ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MANUSCRIPT-BASED THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

BY
Rafael MENELAU OLIVEIRA E CRUZ

DYNAMIC SELECTION OF ENSEMBLE OF CLASSIFIERS USING META-LEARNING

MONTREAL, JUNE 9, 2016

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

Dr. Robert Sabourin, thesis director
Département de génie de la production automatisée - École de Technologie Supérieure

Dr. George D. C. Cavalcanti, co-advisor
Centro de Informática - Universidade Federal de Pernambuco

Dr. Christian Desrosiers, committee president
Département de génie logiciel et des TI - École de Technologie Supérieure

Dr. Laurent Heutte, external examiner
Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes (LITIS) - Université de Rouen

Dr. Ismail Ben Ayed, invited examiner
Département de génie de la production automatisée - École de Technologie Supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON MAY 13, 2016

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

First of all, I would like to express my immense gratitude to my supervisors, Prof. Robert Sabourin and Prof. George Darmiton da Cunha Cavalcanti.

My gratitude also goes to members of my thesis committee: Prof. Laurent Heutte, Prof. Christian Desrosiers and Prof. Ismail Ben Ayed for evaluating this thesis and providing constructive comments.

Many thanks to all my colleagues at LIVIA (Laboratoire d'imagerie, de vision et d'intelligence artificielle) for their help throughout this research. Special thanks to my friends Eduardo Vellasques, Christophe Pagano, Paulo Radtke, Fabio Dittrich, Luiz Gustavo, Idrissa Coulibaly, Luana Batista, Francis Quintal-Lauzon, George Eskanders and Miguel de la Torre, who have always been supportive friends and for helping me to quickly adapt to this new environment.

Most of all thanks to my family who always encouraged me throughout these years. This thesis would not exist without their support.

Finally I would like to the Natural Sciences and Engineering Research Council of Canada (NSERC), the École de technologie supérieure (ÉTS Montréal) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for the financial support.

MÉTA-APPRENTISSAGE POUR LA SÉLECTION DYNAMIQUE DES ENSEMBLES DE CLASSIFIEURS

Rafael MENELAU OLIVEIRA E CRUZ

RÉSUMÉ

Les systèmes de sélection dynamique des ensembles de classifieurs fonctionnent en estimant le niveau de compétence de chaque classifieur estimé dans une région de compétence. Seuls les plus compétents sont choisis dynamiquement pour classer chaque échantillon de test. Le niveau de compétence des classifieurs est généralement estimé à partir du voisinage de l'échantillon à classer, selon un critère donné, comme la performance locale ou la confiance du classifieur de base, calculée sur ce voisinage. Cependant, en utilisant un seul critère de sélection, cela peut conduire à une mauvaise estimation de la compétence du classifieur et par conséquent, sélectionner des classifieurs incompétents.

Dans cette thèse, le mécanisme de sélection dynamique d'un classifieur est formulé comme un méta-problème. Les méta-caractéristiques permettant de représenter ce méta-problème sont les différents critères utilisés normalement pour mesurer le niveau de compétence du classifieur de base. Chaque méta-caractéristique capture une propriété différente du comportement du classifieur de base, et peut être considéré comme un critère différent pour estimer le niveau de compétence d'un classifieur de base telles que la performance de classification dans une région locale de l'espace de caractéristiques et de la confiance du classifieur pour la classification de l'échantillon d'entrée. Ainsi, plusieurs critères peuvent être utilisés conjointement pour une meilleure estimation des compétences des classifieurs.

Dans le chapitre 2, une nouvelle technique de sélection dynamique des ensemble de classifieurs utilisant le méta-apprentissage est proposé, appelé META-DES. Cinq ensembles distincts de méta-caractéristiques, chacun correspondant à un critère différent pour mesurer le niveau de compétence d'un classifieur pour la classification des échantillons d'entrée sont introduits pour ce méta-problème. Les méta-caractéristiques sont extraites des données de validation et utilisées pour entraîner un méta-classifieur pour prédire le niveau de compétence des classifieurs étant donné un exemple à classer. Au cours de la phase de généralisation, les méta-caractéristiques sont extraites de l'instance de requête et transmisent en entrée du méta-classifieur, lequel détermine si un classifieur de base est assez compétent pour être ajouté à l'ensemble. Des expériences sont menées sur plusieurs problèmes de reconnaissance. Les résultats expérimentaux montrent que le META-DES améliore considérablement la performance en classification lorsqu'on les compare à l'état de l'art dans le domaine de la sélection dynamique.

Une analyse étape par étape de chaque processus du système META-DES est présentée au chapitre 3. Nous montrons comment chaque ensemble de méta-caractéristiques est extrait, ainsi que leur impact sur l'estimation du niveau de compétence du classifieur de base. En

outre, une analyse de l'impact de plusieurs facteurs sur la performance du système est réalisée sur le problème synthétique P2 : par exemple, le nombre de classifieurs inclus dans le bassin, de même que la taille des données de validation sont considérés. Les résultats expérimentaux montrent que la sélection dynamique de classifieurs à fonctions discriminantes linéaires à travers le schéma META-DES, permet de résoudre les problèmes de classification caractérisés par une frontière de décision de géométrie complexe.

Dans le chapitre 4, une nouvelle version du schéma META-DES optimisé en fonction de la performance de l'Oracle, appelé META-DES.Oracle est proposée. L'Oracle est une méthode abstraite qui représente un mécanisme de sélection de classifieur idéal. Une sélection de méta-caractéristiques effectuée à l'aide d'une optimisation par essaims particulaires (OEP, ou PSO en anglais) est proposée pour améliorer la performance du méta-classifieur. La différence entre les résultats obtenus par le méta-classifieur et ceux présentés par Oracle est minimisé. L'objectif visé est d'augmenter la performance en sélection du méta-classifieur pour approcher celle de l'Oracle. Les expériences réalisées à l'aide de 30 problèmes de classification démontrent que la procédure d'optimisation basée sur la performance de l'Oracle conduit à une amélioration significative de la précision de la classification par rapport aux versions précédentes du META-DES.

Enfin, au chapitre 5, deux techniques sont analysées afin d'améliorer la performance en généralisation du META-DES, ainsi que des autres techniques de sélection dynamique proposées dans la littérature. Tout d'abord, une technique de sélection de prototypes est appliquée sur les données de validation pour réduire la quantité de chevauchement entre les classes. Au cours de la généralisation, un algorithme K-plus proches voisins adaptatif est utilisé pour une meilleure définition du voisinage de l'échantillon d'essai. Le but visé est d'améliorer l'estimation du niveau de compétence des classifieurs dans la région de compétence en donnant plus d'importance aux exelples qui sont éloignés de la frontière entre les classes. Des expériences ont été effectuées en utilisant 10 techniques de sélection et plus de 30 problèmes de classification. Les résultats démontrent que l'utilisation conjointe de la sélection de prototypes pour éditer des données de validation et la distance d'adaptation locale améliorent sensiblement la précision de la classification des techniques de sélection dynamique.

Mots clés: Reconnaissance de formes, ensemble de classifieurs, sélection dynamique de classifieurs, méta-apprentissage, optimisation par essaims de particles, classifieurs linéaires, perceptrons

DYNAMIC SELECTION OF ENSEMBLE OF CLASSIFIERS USING META-LEARNING

Rafael MENELAU OLIVEIRA E CRUZ

ABSTRACT

Dynamic ensemble selection systems work by estimating the level of competence of each classifier from a pool of classifiers. Only the most competent ones are selected to classify each specific test sample. The classifiers' competences are usually estimated over the neighborhood of the test sample, according to a given criterion, such as the local accuracy estimates or the confidence of the base classifier, computed over the neighborhood of the test sample. However, using only one selection criterion can lead to poor estimation of the classifier's competence. Consequently, the system end up not selecting the most appropriate classifier for the classification of the given test sample.

In this thesis, dynamic ensemble selection is formalized as a meta-problem. From a meta-learning perspective, the dynamic ensemble selection problem is considered as another classification problem, called the meta-problem. The meta-features of the meta-problem are the different criteria used to measure the level of competence of the base classifier. Each set captures a different property of the behavior of the base classifier, and can be seen as a different criterion for estimating the competence level of a base classifier; such criteria include, the classification performance in a local region of the feature space and the classifier confidence for the classification of the input sample. The meta-classifier is trained, based on the defined set of meta-features, to predict the competence level of a given base classifier for the classification of a new test sample. Thus, several criteria can be used in conjunction for a better estimation of the classifiers' competences.

In Chapter 2, a novel dynamic ensemble selection framework using meta-learning is proposed, called META-DES. Five distinct sets of meta-features, each corresponding to a different criterion for measuring the level of competence of a classifier for the classification of input samples are introduced for this specific meta-problem. The meta-features are extracted from the training data and used to train a meta-classifier to predict whether or not a base classifier is competent enough to classify an input instance. During the generalization phase, the meta-features are extracted from the query instance and passed down as input to the meta-classifier. The meta-classifier estimates whether a base classifier is competent enough to be added to the ensemble. Experiments are conducted over several small sample size classification problems, i.e., problems with a high degree of uncertainty due to a lack of training data. Experimental results show the proposed meta-learning framework greatly improves classification accuracy when compared against current state-of-the-art dynamic selection techniques.

In Chapter 3, a step-by-step analysis of each phase of the META-DES framework is conducted. We show how each set of meta-features is extracted as well as their impact on the estimation

of the competence level of the base classifier. Moreover, an analysis of the impact of several factors on the system performance is carried out; these factors include, the number of classifiers in the pool, the use of different linear base classifiers, as well as the size of the validation data. Experimental results demonstrate that using the dynamic selection of linear classifiers through the META-DES framework, it is possible to solve complex non-linear classification problems using only a few linear classifiers.

In Chapter 4, a novel version of the META-DES framework based on the formal definition of the Oracle, called META-DES.Oracle is proposed. The Oracle is an abstract method that represents an ideal classifier selection scheme. A meta-feature selection scheme using an over-fitting cautious BPSO is proposed for improving the performance of the meta-classifier. The difference between the outputs obtained by the meta-classifier and those presented by the Oracle is minimized. Thus, the meta-classifier is expected to provide results that are similar to those of the Oracle. Experiments carried out using 30 classification problems demonstrate that the optimization procedure based on the Oracle definition leads to a significant improvement in classification accuracy when compared to previous versions of the META-DES framework.

Finally, in Chapter 5, two techniques are investigated in order to improve the generalization performance of the META-DES framework as well as any other dynamic selection technique. First, a prototype selection technique is applied over the validation data to reduce the amount of overlap between the classes, producing smoother decision boundaries. During generalization, a local adaptive K-Nearest Neighbor algorithm is employed for a better definition of the neighborhood of the test sample. Thus, DES techniques can better estimate the classifiers' competences. Experiments were conducted using 10 state-of-the-art DES techniques over 30 classification problems. The results demonstrate that the use of prototype selection in editing the validation data and the local adaptive distance significantly improve the classification accuracy of dynamic selection techniques.

Keywords: Ensemble of classifiers, dynamic ensemble selection, classifier competence, meta-learning, particle swarm optimization, linear classifiers

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Page

LIST OF ABREVIATIONS

MCS	Multiple Classifier System
EoC	Ensemble of Classifier
DES	Dynamic Ensemble Selection
DCS	Dynamic Classifier Selection
LCA	Local Classifier Accuracy
OLA	Overall Local Accuracy
MLA	Modified Local Accuracy
DCEID	Dynamic Classifier Ensemble for Imbalanced Data
MCB	Multiple Classifier Behaviour
CSDS	Class-Strength Dynamic Selection
ADS	Ambiguity-guided Dynamic Selection
KNOP	K-Nearests Output Profiles
KNORA	K-Nearests Oracles
KNORA-E	K-Nearests Oracles-Eliminate
KNORA-U	K-Nearests Oracles-Union
DES-FA	Dynamic Ensemble Selection by Filter and Adaptive Distance
DOCS	Dynamic Overproduction and Choose
KLD	Kullback-Leibler Divergence
PSO	Particle Swarm Optimization

BPSO	Binary Particle Swarm Optimization
GV	Global Validation
CD	Critical Difference
RRC	Randomized Reference Classifier
MLP	Multi-Layer Perceptron
SVM	Support Vector Machine
RF	Random Forest
K-NN	K-Nearest Neighbors
KNN	K-Nearest Neighbors
AKNN	Adaptive K-Nearest Neighbors
ENN	Edited Nearest Neighbor
CPS	Classifier Projection Space
RPROP	Resilient Backpropagation
MDS	Multidimensional Scaling
FS	Feature set
MAT	Medial Axial Transformation
HVQ	Hierarchical Vector Quantization
MDF	Modified Directional Features
RBF	Radial Basis Function
NN	Neural Network

KEEL	Knowledge Extraction based on Evolutionary Learning
LKC	Ludmila Kuncheva Collection of real medical data
GA	Genetic Algorithm
DE	Differential Evolution
ACO	Ant Colony Optimization
SFS	Sequential Feature Selection
MBPSO	Modified Binary Particle Swarm Optimization
PV	Partial Validation
BV	Backwarding Validation
MD	Minimum difference
RC	Random Classifier

LISTE OF SYMBOLS AND UNITS OF MEASUREMENTS

λ	Meta-Classifier
$\mathcal{T}$	Training dataset
$\mathcal{T}_\lambda$	Meta-training dataset
$DSEL$	Dynamic selection dataset
$\mathcal{G}$	Generalization dataset
$\mathcal{T}_\lambda^*$	$\mathcal{T}_\lambda^*$ transformed into meta-data
c_i	i-th classifier
C	Pool of classifiers
C'	Ensemble of classifiers
C^*	A population of EoC
M'	Size of the population of EoC
$\mathbf{x}_j$	j-th sample
$\tilde{\mathbf{x}}_j$	Output profile of the j-th sample
M	Size of the pool of classifiers C
N	Number of instances in a dataset
C'	Ensemble of classifiers
f_i	i-th meta-feature set
$P(w_l \mid \mathbf{x}_j)$	Posterior probability obtained for the sample $\mathbf{x}_j)$
$P(\mathbf{x}_j)$	Prior probability

$v_{i,j}$	meta-feature vector of the base classifier c_i for the sample $\mathbf{x}_j$
$\alpha_{i,j}$	Class label of the meta-feature vector $v_{i,j}$
$\delta_{i,j}$	Competence level of the classifier c_i for the sample $\mathbf{x}_j$
Ω	Set of class labels
L	Number of class labels
w_l	The l-th class in the set ω
$H\left(\mathbf{x}_{j,train_\lambda}, C\right)$	Degree of consensus among the pool of classifiers C
h_c	Consensus threshold
θ_j	Region of competence of the j-th instance
K	Number of samples in the region of competence
ϕ_j	Closest output profiles of the j-th instance
K_P	size of ϕ_j
λ_D	Meta-Classifier problem-dependent
λ_I	Meta-Classifier problem-independent
λ_{ALL}	Meta-Classifier hybrid
Λ	The set of meta-classifiers
ρ	Pearson's correlation coefficient
$\mathcal{D}_i$	A single classification problem
$\mathcal{D}$	The set of classification problems
Υ	Competence threshold

$\tilde{D}_{SEL}$	Output profiles of the dynamic selection dataset
$DSEL^*$	Meta-data extracted from DSEL
$\mathcal{T}_{\lambda}^{T}$	Meta-tarining dataset
$\mathcal{T}_{\lambda}^{O}$	Meta-optimization dataset
$DSEL^*$	Meta-data extracted from DSEL.
$pBest_i$	Best position visited by the i-th particle
$gBest$	Global best position considering the whole swarm
g	Generation of the BPSO algorithm
$max(g)$	Maximum number of generation used in the BPSO algorithm
$\mathcal{S}$	Swarm of particles
$\mathcal{S}_i$	Particle (Solution) in the BPSO algorithm
$velocity_i$	Velocity of the i-th particle
c_1, c_2	Acceleration coefficients
$d_{\lambda,Oracle}$	Distance between the meta-classifier λ and the Oracle in the dissimilarity space
$\mathscr{T}_S(x)$	S-shaped transfer function
$\mathscr{T}_V(x)$	V-shaped transfer function
$max(S)$	Number of particles S
$\mathscr{A}$	Particle Archive
$S(\mathbf{x}_j)$	Vector of class supports obtained for the sample $\mathbf{x}_j$
$S_{lk}(\mathbf{x}_j)$	Support given for the correct class label of $\mathbf{x}_j$

q_α	Critical value in the Bonferroni-Dunn test
c	regularization parameter
γ	Kernel spread parameter
DSEL$'$	The edited Dynamic selection dataset
$R_{j,DSEL'}$	Radius of the j-th hypersphere
$d_{adaptive}$	The adaptive K-NN distance
H_0	Null hypothesis
n_{exp}	Number of experiments
α	Significance level
n_c	Critical Value for the Sign test
$d(i,j)$	Dissimilarity between the classifiers C_i and C_j
$\mathcal{S}$	Stress function
D	Dissimilarity matrix
$\tilde{D}$	Projection of the dissimilarity matrix D onto the 2-dimensional CPS
F_i	A feature representation
R_x, R_y	Horizontal and vertical Robert operators
S_x, S_y	Horizontal and vertical Sobel operators
I_{m_x}, I_{m_y}	X-gradient and Y-gradient images
$\Theta(i,j)$	Phase of the pixel ij
$r(i,j)$	Magnitude of the pixel ij

INTRODUCTION

In recent years, Multiple Classifier Systems(MCS) have been widely studied as an alternative for increasing efficiency and accuracy in pattern recognition applications. Several theoretical and empirical studies have shown that a multiple classifier system or an ensemble of classifiers produces more accurate recognition performance than a single classifier.

Generally speaking, MCS are composed of three stages [1]: (1) Generation, (2) Selection, and (3) Integration or fusion (Figure 0.1). In the generation phase, a pool of classifiers is trained. Here, the main concern in this training stage is creating a pool of classifier that are diverse and accurate. Diversity in the context of MCS is related to the errors committed by different classifiers. A pool of classifiers is considered diverse if its members make wrong predictions in different instances. Consequently, combining their decisions is likely to improve the classification accuracy.

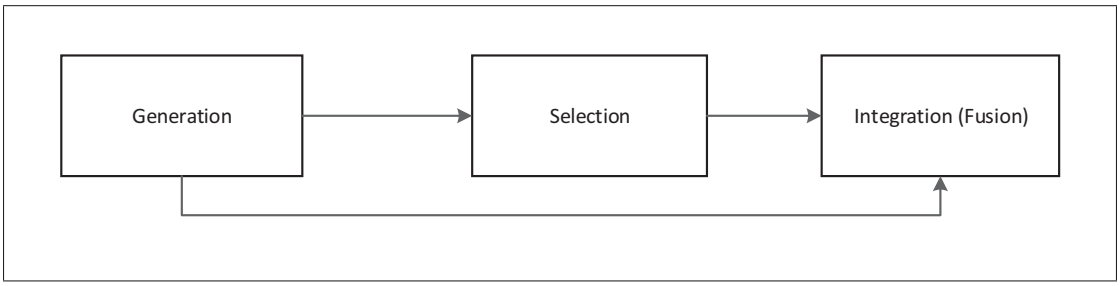


Figure 0.1 Three phases of a MCS [Adapted from [1]]

Several methods have been proposed to obtain a pool of diverse classifiers, such as Bagging [3], Random Subspace [4], AdaBoost [5], and Random Linear Oracles [6]. Diversity can also be achieved by using different learning algorithms in the pool, such as Support Vector Machines (SVM), Multi-Layer Perceptron neural networks (MLP), Decision Trees (DT), etc. In some applications, such as biometrics, a pool of diverse classifiers can be obtained by using entirely different feature domains, e.g., identification by speech and image [7]. A comprehensive analysis of diversity and its implications in multiple classifier systems is presented in [8; 9].

In the second stage, based on a pool of classifiers, the goal is to select a subset containing the most competent ones. In this thesis, we refer to the selected subset of classifiers as an Ensemble of Classifiers (EoC). The classifier selection stage can be subdivided into two groups: static and dynamic. In static approaches [10; 11; 12; 13], the selection is performed during the training stage of the system, considering the global performance of the base classifiers over either the training or the validation dataset. The selected classifier or EoC is then used to classify of all unseen test samples. By contrast, dynamic ensemble selection approaches (DES) [14; 15; 16; 17; 18; 19; 20; 21; 22; 23] select a different classifier or a different EoC for each new test sample.

The third stage of an MCS consists in combining the outputs of the selected classifiers to predict the label of a given test instance. There are several techniques for combining of multiple classifiers, such as probabilistic models based on the posterior probabilities obtained by the base classifiers as presented in [24], Decision Templates and Dempster-Shafer combination [25]. Further, other classifier, such as a gating networks in mixture of experts [26], can be trained to integrate the output of the base classifiers.

Problem Statement

This thesis is focused on the selection stage, more specifically on dynamic classifier and ensemble selection techniques. DES techniques rely on the assumption that each base classifier is an expert in a different local region of the feature space [27]. Thus, given a new test sample, DES techniques aim to select the most competent classifiers for the local region in the feature space where the test sample is located. Only the classifiers that attain a certain competence level, according to a selection criterion, are selected. Recent work in the dynamic selection literature demonstrates that dynamic selection techniques constitute an effective tool for classification problems that are ill-defined, i.e., for problems where the size of the training data is small and there are not enough data available to model the classifiers [16; 17].

The key issue encountered in DES is to define the criterion for measuring the level of competence of a base classifier. Most DES techniques [14; 22; 21; 20; 28; 29; 30; 31] use estimates of the classifiers' local accuracy in small regions of the feature space surrounding the query instance as a search criterion for performing the ensemble selection. On the other hand, DES techniques based on other criteria, such as the degree of consensus of the ensemble classifiers [15; 16], encounter some problems when the search cannot find consensus among the ensembles. In addition, they neglect the local performance of the base classifiers.

A crucial concept in the DES literature is the definition of the Oracle. The Oracle is an abstract model defined in [32], which always selects the classifier that predicted the correct label, for the given query sample, if such a classifier exists. In other words, it represents the ideal classifier selection scheme. The Oracle is used in the DES literature in order to determine whether the results obtained by proposed DES techniques is close to ideal accuracy or whether they leave room for improvement. As reported in a recent survey [1], the results obtained by DES techniques based solely on one source of information are still far from those achieved by the Oracle. In addition, as stated by Ko et al. [14], addressing the behavior of the Oracle is much more complex than applying a simple neighborhood approach, and the task of figuring out its behavior based merely on a single source of information is not an easy one. Thus, multiple sources of information should be taken into account in order to improve the classification accuracy of DES techniques, and to achieve results closer to the Oracle.

Furthermore, as stipulated by the "No Free Lunch" theorem [33], no algorithm is better than any other over all possible classes of problems. Using a single criterion to measure the level of competence of a base classifier is highly likely to lead to erroneous results. In our opinion, the information captured by the different criteria reflects different properties of the behavior of a base classifier, and we believe that these properties can be complementary. Thus, multiple criteria should be taken into account in measuring the competence level of a base classifier in order to achieve a more robust dynamic ensemble selection technique.

objectives

The objective of this thesis is to develop a framework in which several sources of information or criteria can be used to obtain a better estimates of the classifier competences for dynamic selection. In addition, since distinct classification problems are associated with different degrees of difficulty and data complexity [34], this thesis proposes a framework which also adapt to the intrinsic characteristics of each classification problem by selecting the most appropriate criteria for conducting the dynamic selection of classifiers.

The desired property is achieved by defining dynamic ensemble selection as a meta-problem. From a meta-learning perspective, the dynamic ensemble selection problem is considered as another classification problem, called the meta-problem. The meta-features of the meta-problem are the different criteria used to measure the level of competence of the base classifier. Each set captures a different property of the behavior of the base classifier, and can be seen as a different criterion for estimating the competence level of a base classifier; such criteria include the classification performance in a local region of the feature space and the classifier confidence for the classification of the input sample.

The meta-features are used as input to a meta-classifier that decides whether or not a base classifier is competent enough for the classification of an input sample based on the predefined meta-features. When distinct criteria, encoded as meta-features, are used, although one criterion might fail to properly estimate the competence level of the base classifier, due to a high presence of noise in some local regions of the feature space [20] or due to low confidence results [35], the system can still achieve good performance, since other meta-features are also considered by the selection scheme. Hence, the expectation is that the proposed framework can obtain higher classification accuracy.

Contributions

The main contribution of this thesis is its formalization of dynamic ensemble selection as a meta-problem, leading to the proposal of a novel dynamic ensemble selection framework using meta-learning, called META-DES.

Since this thesis is manuscript-based, each chapter presents a different contribution to the development of the META-DES framework, as well as means of improving dynamic ensemble selection techniques in general. The contributions are listed below:

- In Chapter 2, the dynamic ensemble selection problem is formalized as a meta-problem and the META-DES framework is presented. Five sets of meta-features are proposed. Experimental results demonstrates that the META-DES outperforms the current state-of-the-art dynamic classifier and ensemble selection techniques in several classification benchmarks.
- In Chapter 3, a deep analysis of each phase of the META-DES framework is conducted using synthetic datasets in order to provide a better understanding of the framework. The impact of each set of meta-feature for the estimation of the competence level of the base classifiers is conducted.
- In Chapter 4, an optimization procedure for the META-DES framework is presented. The optimization scheme is guided by the formal definition of the Oracle, which is an abstract method that represents the ideal dynamic selection technique. The optimization procedure significantly improves the classification performance of the META-DES framework.
- In Chapter 5, the influence of the data distribution in the dynamic selection dataset is analyzed. A prototype selection and adaptive distance are proposed for improving the performance of DES techniques. It is important to mention that the contribution of this chapter can be generalized to any dynamic classifier and ensemble selection technique.

Furthermore, in Appendix III a static classifier selection scheme using the definition of the Oracle is presented for the problem of handwriting recognition.

Organization of the thesis

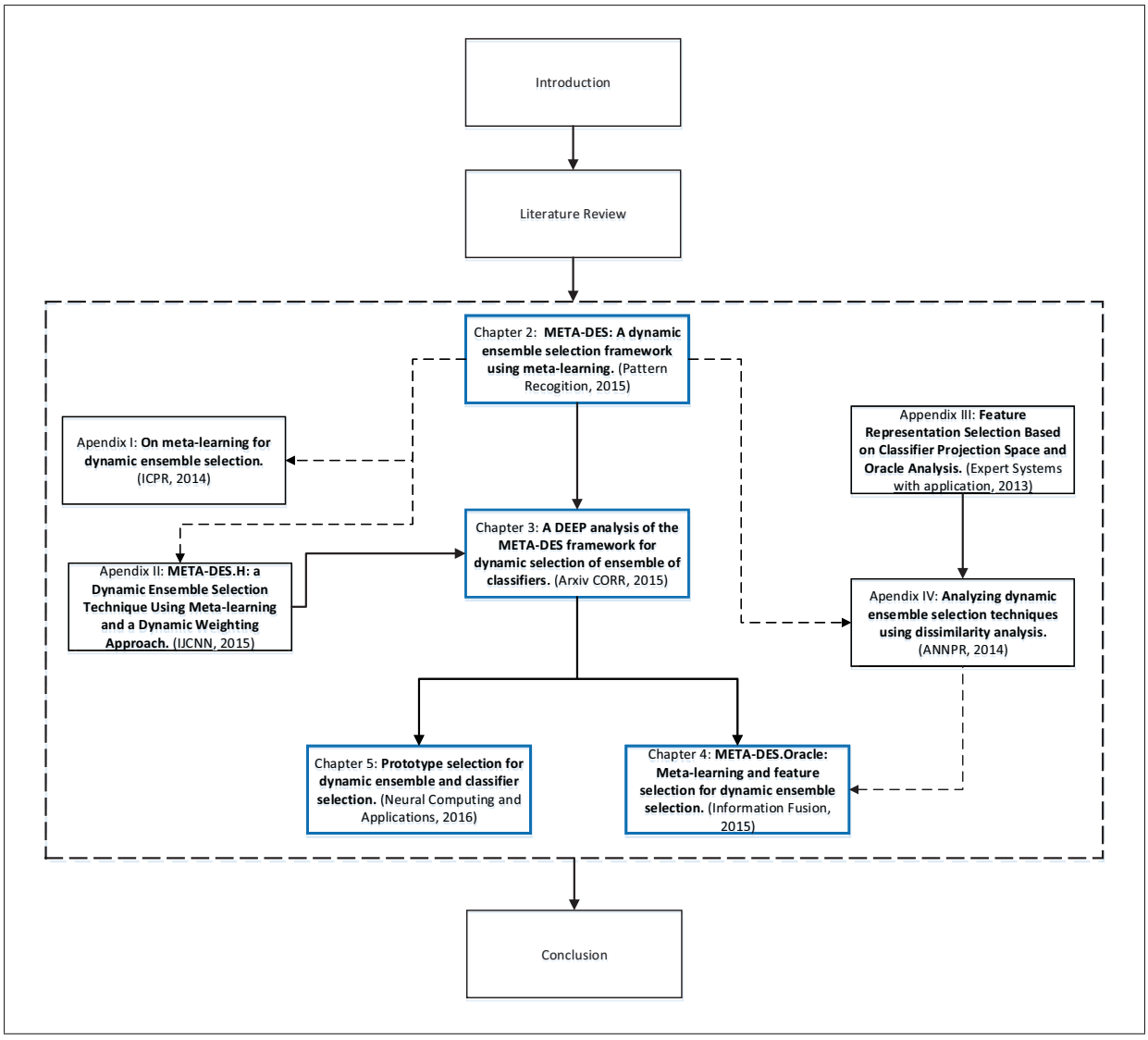


Figure 0.2 The flow of the thesis is shown by connected boxes. The solid arrows indicate the dependencies between the chapters and appendixes (i.e., one chapter must be read beforehand). Dashed arrows indicates the suggested readings between the chapters and appendices for a better comprehension

This manuscript-based thesis is organized into five chapters. Figure 0.2 presents an overview of the organization of this thesis. The chapters and appendixes inside the dashed box represent the articles that were published or submitted during the development of the thesis. The solid

arrows indicate the dependencies between the chapters (i.e., one chapter must be read before the other for a better understanding of the proposed techniques). In addition, the dashed arrows indicate the relationship between each chapter and the appendices. The appendices extends the work of the corresponding chapters by exploring different aspects of the framework.

This thesis starts with an overview of dynamic selection techniques and meta-learning presented in the first chapter. They are presented from a classifier competence point of view, which is the main concern examined in this thesis.

In Chapter 2, the dynamic ensemble selection problem is formalized as a meta-problem, and the META-DES framework is presented. The meta-problem consists in defining whether the base classifier is competent enough to classify a given query sample. The meta-features of the meta-problem are the criteria used to measure the level of competence of the base classifier. A total of five sets of meta-features are proposed, each being a different property of the behavior of the base classifier. A meta-classifier is then trained, based on the meta-features, to determine whether a base classifier is competent to predict the label of a given input pattern. The contents of this chapter have been published in the Pattern Recognition journal.

Based on the framework proposed in Chapter 2, three training scenarios for the meta-classifier are evaluated in Appendix I: problem-dependent, problem-independent and hybrid. In the problem-dependent scenario, the meta-classifier is trained using meta-data extracted from one classification problem, and is used as the classifier selector on the same problem. In the problem-independent scenario, the meta-classifier is trained using the meta-data extracted from one classification problem, and is used as the classifier selector on a different one. In the hybrid scenario, a single meta-classifier is trained using the meta-data extracted from all classification problems considered in this work, and is used as the classifier selector for all classification problems. Experimental results demonstrate that the training of the meta-classifier is problem-dependent. Moreover, a strong correlation is found between the performance of the meta-classifier and the classification accuracy of the META-DES. The contents of this ap-

pendix was published in the Proceeding of the International Conference on Pattern Recognition (ICPR) [36].

In Appendix II, two modifications to the META-DES framework are proposed: In the training phase, we evaluate four different algorithms for the training of the meta-classifier. For the generalization phase, a hybrid dynamic selection and weighting approach is proposed. The hybrid scheme works as follows: first, an ensemble with the most competent classifiers is selected. Then, the weights of the selected classifiers are estimated based on their levels of competence. Thus, classifiers that attain the highest level of competence, for the classification of the given query sample, have a greater impact on the final decision. The proposed hybrid approach is called META-DES.H, and since it outperformed the META-DES, it is used in the following chapters of this thesis.

In Chapter 3, a step-by-step analysis of each phase of the framework during training and test is presented in order to provide a better understanding of the META-DES framework as a white box. The influence of each set of meta-features, as well as their impact on the estimation of the competence level of the base classifier are shown. In addition, an analysis of the impact of several factors of the system performance, such as the number of classifiers in the pool, the use of different linear base classifiers, as well as the size of the validation data are also presented. Two types of linear classifiers, Perceptron and Decision Stumps, are considered. We show that using the dynamic selection of linear classifiers through the META-DES framework, we can solve complex non-linear classification problems where static combination techniques such as AdaBoost cannot. The conclusions from this analysis servers as guidelines for further developments in the META-DES framework, as well as for the understanding of dynamic ensemble selection techniques in general. The contents of this chapter have been published in arXiv Computing Research Repository (CORR) [37].

In Chapter 4, a novel version of the META-DES framework based on the formal definition of the Oracle, called META-DES.Oracle, is proposed. First, 15 sets of meta-features are proposed. Then, a meta-feature selection scheme using an overfitting cautious BPSO is proposed

for minimizing the difference between the outputs obtained by the meta-classifier and those obtained by the Oracle. Thus, the meta-classifier can present results that are similar to those of the Oracle. Experiments conducted over 30 classification datasets demonstrate that the META-DES.Oracle outperforms both the META-DES and META-DES.H. In addition, this chapter also demonstrates that all 15 sets of meta-features are relevant in addressing the complex behavior of the Oracle, and that the choice of the best set of meta-features varies considerably according to different classification problems. The contents of this chapter have been submitted to the Information Fusion journal.

The idea behind the optimization scheme proposed in Chapter 4 derives from the analysis conducted in Appendix III and IV. In Appendix III, a new framework for analyzing the relationship between different feature representations is proposed. The Oracle definition is used to select the best subset of feature representations for the problem of handwritten digits and character recognition. The contents of this appendix were published in the Expert Systems With Applications journal [11].

In Appendix IV, the dissimilarity analysis presented in Appendix III is conducted in order to understand the relationship between different criteria used in the literature with the dynamic selection techniques to estimate the competence level of the base classifiers. The behavior of the Oracle is also studied in the dissimilarity analysis framework. The dissimilarity analysis shows that using meta-learning to combine several DES criteria is more likely to present a behavior closer to the Oracle in the dissimilarity space. In addition, techniques that appear closer to the Oracle in the dissimilarity space are more likely to achieve better classification accuracy.

Chapter 5 comes directly from the analysis conducted in Chapter 3, showing that the distribution of the dynamic selection dataset (DSEL), which is used to compute the competence of the base classifiers during generalization, has a huge influence on the performance of the DES techniques. In this chapter we demonstrate, using synthetic data, that the proposed META-DES technique may not produce consistent results when there is a high degree of noise in DSEL, and

so two techniques are therefore proposed: in the training stage, a prototype selection mechanism is applied in DSEL to eliminate instances with a high risk of being noise and also to reduce the amount of overlap between the classes. During the generalization stage, the local regions of the query sample are estimated using an adaptive KNN rule (A-KNN), which shifts the region of competence from the class border to the class centers. As such, samples that are more likely to be noise are less likely to be selected to compose the region of competence. The proposed method can be applied to any dynamic selection technique that uses local information in estimating the competence of the base classifier. Experimental results demonstrate that the proposed scheme significantly improves the classification performance of several DCS and DES techniques, including the META-DES framework. The contents of this chapter have been submitted to the Neural Computing and Applications journal.

Finally, a general conclusion and future works are presented in the last chapter of this thesis.

CHAPTER 1

DYNAMIC CLASSIFIER AND ENSEMBLE SELECTION REVIEW

Dynamic selection techniques consist, based on a pool of classifiers C, in finding a single classifier c_i, or an ensemble of classifiers C', having the most competent classifiers to predict the lass label for a specific instance, $\mathbf{x}_j$. Recent works in classifier and ensemble selection have shown a preference for dynamic ensemble selection over static ensemble selection, especially in dealing with ill-defined problems, i.e., when the size of the dataset is small and there is not enough data to train a strong classifier having a lot of parameters to learn [16]. In addition, due to insufficient training data, the distribution of the training data may not adequately represent the real distribution of the problem. Consequently, the classifiers cannot learn the separation between the classes in those cases.

The rationale behind dynamic ensemble selection techniques resides in the observation that not every classifier in the pool is an expert in classifying all unknown samples. Each base classifier is an expert in a different local region of the feature space [27]. Moreover, different patterns are associated with distinct degrees of difficulties. It is therefore reasonable to assume that only a few base classifiers can predict the correct class label.

Early works in dynamic selection started with the selection of a single classifier rather than an EoC. In such techniques, only the classifier that attained the highest competence level is used for the classification of the given test sample. These techniques are called dynamic classifier selection (DCS). The local classifier accuracy (LCA) [22] and the multiple classifier behavior (MCB) [21] are examples of DCS techniques. However, given the fact that selecting only one classifier can be very error prone, some researchers decided to select a subset of the pool of classifiers, C, containing all classifiers that attained a certain competence level, rather than a single model. Such techniques are called dynamic ensemble selection (DES). Example of DES techniques are the K-Nearests Oracles (KNORA) [14], K-Nearests Output Profiles (KNOP) [16], Dynamic Overproduce and Choose (DOCS) [15], and the method based on Randomized Reference Classifier (RRC) DES-RRC [18].

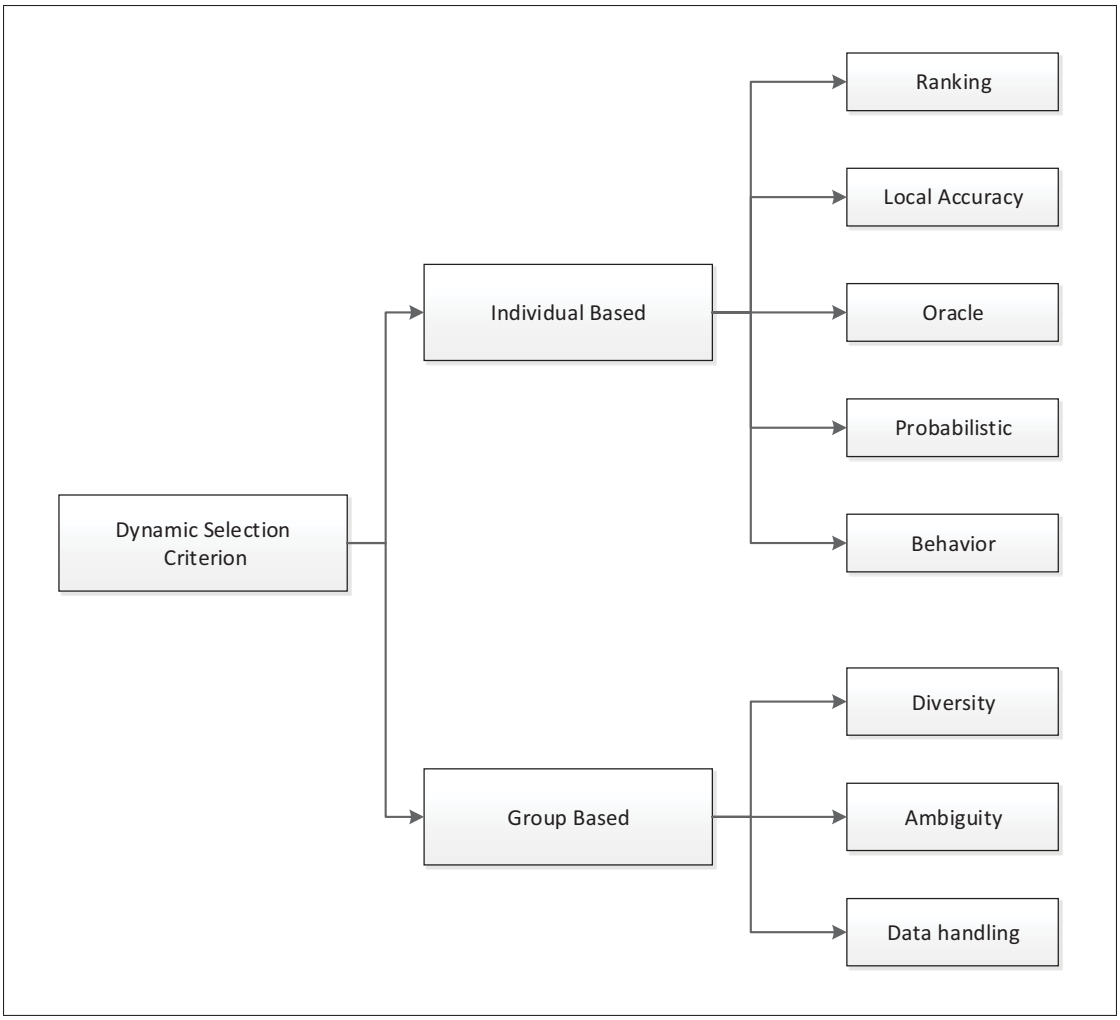


Figure 1.1 Taxonomy of the criteria for estimating the competence level in dynamic selection [Adapted from [1]]

The key issue underlying dynamic selection is the notion of competence. The competence level of a classifier defines how much we trust the expert, for the given classification task. The most important component of dynamic selection techniques is the criterion used to measure the competence level of the base classifiers, given a specific test sample $\mathbf{x}_j$. The most common approach in the literature involves estimating the competence of the base classifiers in small regions of the feature space surrounding the query sample, $\mathbf{x}_j$. This local region is usually defined based on the KNN technique applied to either the training [22] or validation data [14]. In this thesis, we refer to such a set as the dynamic selection dataset, DSEL, in line with recent works in the DES literature [1; 16; 17].

The set with the K-Nearest Neighbors of a given test sample $\mathbf{x}_j$ is called the region of competence, and is denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. Usually, the samples belonging to θ_j are used to estimate the competence of the base classifiers, based on various criteria, for the classification of an unseen sample $\mathbf{x}_j$.

The criteria can be organized into two groups (Figure 1.1): individual-based and group-based measures. The former present the measures where the individual performance of the base classifier is used to estimate its level of competence. This category can be further divided into five subgroups [1]: Ranking [22; 38], Accuracy [22; 20], Probabilistic Models [18; 39; 40; 41], Behavior [21; 16] and Oracle [14].

The group-based measures are composed of metrics that take into account the interaction between the classifiers in the pool. This category can be further divided into three subgroups [1]: Diversity [42; 43], Data Handling [19] and Ambiguity [15]. These measures are not directly related to the notion of competence of a base classifier, but to the notion of pertinence, i.e., whether the base classifier work well in conjunction with other classifiers in the ensemble.

In the following sections, the DCS and DES techniques based on each source of information are presented. In Appendix IV, the selection criteria embedded in several classifier and dynamic selection techniques are analyzed from the classifier competence point of view. Furthermore, the pseudo-code for each technique is presented in the following survey [1].

1.1 Individual-based measures

1.1.1 Ranking

Classifier rank was one of the first criteria proposed for estimating the competence level of base classifiers in a dynamic selection. The ranking of a single base classifier c_i could be estimated simply by the number of consecutive correctly classified samples. The classifier that correctly classifies the greatest number of consecutive samples derived from the validation data is considered to have the highest competence level or "rank". In addition, alternative

ranking techniques based on mutual information were also proposed [38], but, because of their complexity and because they were only defined to work with the Nearest Neighbor (NN) as base classifiers, recent work has preferred the use of the simplified ranking method.

1.1.2 Local Accuracy

Classifier accuracy is the most commonly used criterion for dynamic classifier and ensemble selection techniques [22; 14; 20; 30; 23; 28; 38; 29; 19]. Techniques that are based on local accuracy first computes the region of competence. θ_j of the given test sample $\mathbf{x}_j$. The region of competence can be defined either on the training set [22] or on the validation set [14].

Based on the samples belonging to the region of competence, θ_j, different means have been proposed for estimating the local accuracy of the base classifier. For example, the Overall Local Accuracy (OLA) [22] technique uses the accuracy of the base classifier in the whole region of competence as a criterion for measuring its level of competence. The classifier that obtains the highest accuracy rate is considered the most competent. The Local Classifier Accuracy (LCA) [22] computes the performance of the base classifier in relation to a specific class label. The Modified Local Accuracy [29] works similarly to the LCA technique, with the only difference being that each sample belonging to the region of competence is weighted by its Euclidean distance to the query instance. As such, instances from the region of competence that are closer to the test sample have a higher degree of influence when computing the performance of the base classifier. Moreover, variations of the OLA and LCA techniques using *a priori* and *a posteriori* probabilities were proposed by Didaci et al. [44] for obtaining more precise estimates of the competence level of a base classifier.

The difference between these techniques lies in how they utilize the local accuracy information in order to measure the level of competence of a base classifier. The main problem with these techniques is their dependence on the definition of the region of competence, often performed via K-NN or clustering techniques. The dynamic selection technique is likely to commit errors when there is a high degree of overlap between the classes [20]. As reported in [14], using the

local accuracy information alone is not sufficient to achieve results close to the Oracle. Moreover, any difference between the distribution of validation and test datasets may negatively affect the system performance.

1.1.3 Oracle

Oracle-based techniques can be considered as a particular case of local accuracy techniques, where the base classifier is expected to present perfect accuracy in the region of competence. In other words, it can be interpreted as a "local Oracle" [1]. From this perspective, Ko et al. [14] proposed the K-Nearest Oracles (KNORA) family of techniques, inspired by the Oracle concept. Four techniques were proposed. The KNORA-Eliminate (KNORA-E), which considers that a base classifier c_i is competent for the classification of the query instance $\mathbf{x}_j$ if c_i achieves a perfect accuracy for the whole region of competence. Only the base classifiers with a perfect accuracy are used during the voting scheme. In The KNORA-Union (KNORA-U) technique, the level of competence of a base classifier c_i is measured by the number of correctly classified samples in the defined region of competence. In this case, every classifier that correctly classified at least one sample can submit a vote. In addition, two weighted versions, KNORA-E-W and KNORA-U-W were also proposed, in which the influence of each sample belonging to the region of competence is weighted based on its Euclidean distance to the query sample $\mathbf{x}_j$.

1.1.4 Probabilistic

This class of meta-features is based on probabilistic models that are applied over the vector of class supports produced by the base classifier c_i for the classification of a given query sample. The motivation behind probabilistic measures derives from the observation that classifiers that perform worse than the random classifier, i.e., a classifier that randomly selects the classes with equal probabilities, deteriorate the majority voting performance. In contrast, if the base classifiers are significantly better than the random classifier, they are likely to improve the majority voting accuracy [45]. Hence, each source of information belonging to this subgroup estimates

the probability that the decisions of a given base classifier c_i are significantly different from that of a random classifier from different probabilistic or information-theoretic perspectives.

Several probabilistic criteria have been proposed for estimating the level of competence of the base classifier, such as Logarithmic difference [46], Kullback-Leibler divergence [45], and the Randomized Reference Classifier (DES-RRC) [18]. Moreover, techniques based on probabilistic criteria have been successfully applied for the recognition of EMG signals in a bio-prosthetic hand [47].

1.1.5 Behavior

This subgroup is based on information that is computed from the behavior of the predictions made by the pool of classifiers through the concept of output profiles [16]. The output profile of an instance $\mathbf{x}_j$ is denoted by $\tilde{\mathbf{x}}_j = \left\{\tilde{\mathbf{x}}_{j,1}, \tilde{\mathbf{x}}_{j,2}, \ldots, \tilde{\mathbf{x}}_{j,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,i}$ is the decision yielded by the base classifier c_i for the sample $\mathbf{x}_j$. In such techniques, the local regions are defined in the output profiles space, also called decision space, by computing the distances between the output profile of the query sample, and those of the dynamic selection dataset.

Based on the information extracted from the decision space, the K-Nearest Output Profile (KNOP) [17] is similar to the KNORA technique, with the difference being that the KNORA works in the feature space, while the KNOP works in the decision space. The KNOP technique first defines a set with the samples that are most similar to the output profile of the input sample, $\tilde{\mathbf{x}}_j$ in the decision space, called the output profiles set. The validation set is used for this purpose. Then, similarly to the KNORA-E technique, only the base classifiers that achieve a perfect recognition accuracy for the samples belonging to the output profiles set are used during the voting scheme. The Multiple Classifier Behaviour (MCB) technique [21] also defines a set with the most similar output profiles to the input sample using the decision space. Here, the selection criterion is based on a threshold. The base classifiers that achieve a performance higher than the predefined threshold, based on the Behavior-Knowledge Space (BKS) [48], are considered competent, and are selected to predict the class label of the given test sample.

1.2 Group based measures

Group-based methods work by estimating the competence level of a whole ensemble of classifiers rather than each classifier individually. This may be achieved either by generating a population of EoC, $C^* = \{C'_1, C'_2, \ldots, C'_{M'}\}$ (M' is the number of EoC generated), or using an optimization algorithm such as genetic algorithms or greedy search [12; 15; 13]. In addition, some techniques first select an EoC according to some individual-based criterion, such as local accuracy, and then either add or remove classifiers from the selected EoC.

1.2.1 Diversity

Diversity in the context of dynamic selection has been used by some authors as a post-processing means of improving classification performance after an ensemble is selected. Several metrics for measuring diversity in an EoC have been proposed [8; 49]. Of all diversity measures, the Double-Fault [50] measure garnered a lot of attention as it presents a higher correlation with the majority voting accuracy [8] when compared to other diversity measures.

In [42], two DES techniques combining accuracy and diversity, *K-NN and Selection* and *Cluster and Selection*, are proposed. They differ in how the region of competence of a test sample is defined; the former is based on the K-NN technique, while the latter uses the K-Means clustering algorithm. The dynamic selection stages of both techniques are similar. First, the classifiers are sorted based on their classification performance in the region of competence. A predefined number of classifiers with the highest performance are selected to compose the EoC. After that, the most diverse classifiers in relation to the selected EoC are added to the ensemble. In that case, the double fault diversity measure is considered. An empirical comparison of dynamic selection techniques based on accuracy and diversity was conducted by Souto et al. [51].

Another interesting DES technique that uses a diversity measure is the DES-CD method proposed by Lysiak et al. [52]. In the DES-CD methods, the most competent classifiers are selected based on the randomized reference classifier (RRC) proposed in [18]. Then, other classifiers in the pool are added to the selected ensemble if they increase diversity.

1.2.2 Ambiguity

Different ways of measuring the ambiguity or the consensus of the ensemble have been proposed: Margin-based Dynamic Selection (MDS) [15], where the criterion is the margin between the most voted class and the second most voted class. The margin is computed simply by considering the difference between the number of votes received by the most voted class and those received by the second most voted class. Two variations of the MDS were proposed in [15], namely, the Class-Strength Dynamic Selection (CSDS), which includes the ensemble decision in the computation of the MDS, and the GSDS, where the global performance of each EoC is also taken into account [16]. Another technique from this paradigm is the Ambiguity-guided Dynamic Selection (ADS) [15], which uses the ambiguity among the base classifiers of an EoC as the criterion for measuring the competence level of an EoC. The ambiguity is calculated by the number of base classifiers of an ensemble that disagrees with the ensemble decision. The lower this number is, the higher the level of competence of the EoC.

The advantage of ambiguity as a DES criterion stems from the fact that it does not require information from the local region. However, in many cases, these techniques cannot find an EoC with an acceptable confidence level. There is a tie between different members of the pool, and the systems end up performing a random decision [16]. In addition, the pre-computation of ensembles also greatly increases the overall system complexity as we are dealing with a pool of EoC rather than a pool of classifiers.

1.2.3 Data handling

Xiao et al.[53] propose an interesting adaptive ensemble selection approach based on data handling theory (GDMH) and complexity models. The system is based on a multivariate analysis theory for modeling complexity systems presented in [54]. Given a new test sample $\mathbf{x}_j$, several ensemble configurations are evaluated using the GMDH. Then, the ensemble with optimal complexity is selected. Furthermore, a modification of the GDES method for dealing

specifically with imbalanced distributions, called Dynamic Classifier Ensemble Selection with Imbalance Distribution (DCEID), was proposed in [19].

CHAPTER 2

META-DES: A DYNAMIC ENSEMBLE SELECTION FRAMEWORK USING META-LEARNING

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2], Tsang Ing Ren[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Published in « Pattern Recognition » 2015.

Abstract

Dynamic ensemble selection systems work by estimating the level of competence of each classifier from a pool of classifiers. Only the most competent ones are selected to classify a given test sample. This is achieved by defining a criterion to measure the level of competence of a base classifier, such as, its accuracy in local regions of the feature space around the query instance. However, using only one criterion about the behavior of a base classifier is not sufficient to accurately estimate its level of competence. In this paper, we present a novel dynamic ensemble selection framework using meta-learning. We propose five distinct sets of meta-features, each one corresponding to a different criterion to measure the level of competence of a classifier for the classification of input samples. The meta-features are extracted from the training data and used to train a meta-classifier to predict whether or not a base classifier is competent enough to classify an input instance. During the generalization phase, the meta-features are extracted from the query instance and passed down as input to the meta-classifier. The meta-classifier estimates, whether a base classifier is competent enough to be added to the ensemble. Experiments are conducted over several small sample size classification problems, i.e., problems with a high degree of uncertainty due to the lack of training data. Experimental results show the proposed meta-learning framework greatly improves classification accuracy when compared against current state-of-the-art dynamic ensemble selection techniques.

2.1 Introduction

Multiple Classifier Systems (MCS) aim to combine classifiers to increase the recognition accuracy in pattern recognition systems [24; 9]. MCS are composed of three phases [1]: (1) Generation, (2) Selection and (3) Integration. In the first phase, a pool of classifiers is generated. In the second phase, a single classifier or a subset having the best classifiers of the pool is(are) selected. We refer to the subset of classifiers as Ensemble of Classifiers (EoC). The last phase is the integration, and the predictions of the selected classifiers are combined to obtain the final decision [24].

For the second phase, there are two types of selection approaches: static and dynamic. In static approaches, the selection is performed during the training stage of the system. Then, the selected classifier or EoC is used for the classification of all unseen test samples. In contrast, dynamic ensemble selection approaches (DES) [14; 15; 16; 17; 18; 19; 20; 21; 22; 23] select a different classifier or a different EoC for each new test sample. DES techniques rely on the assumption that each base classifier is an expert in a different local region of the feature space [27]. So, given a new test sample, DES techniques aim to select the most competent classifiers for the local region in the feature space where the test sample is located. Only the classifiers that attain a certain competence level, according to a selection criterion, are selected. Recent work in the dynamic selection literature demonstrates that dynamic selection techniques is an effective tool for classification problems that are ill-defined, i.e., for problems where the size of the training data is small and there are not enough data available to model the classifiers [16; 17].

The key issue in DES is to define a criterion to measure the level of competence of a base classifier. Most DES techniques [14; 22; 21; 20; 28; 29; 30; 31] use estimates of the classifiers' local accuracy in small regions of the feature space surrounding the query instance as a search criterion to perform the ensemble selection. However, in our previous work [20], we demonstrated that the use of local accuracy estimates alone is insufficient to achieve results close to the Oracle performance. The Oracle is an abstract model defined in [32] which always selects

the classifier that predicted the correct label, for the given query sample, if such classifier exists. In other words, it represents the ideal classifier selection scheme. In addition, as reported by Ko et al. [14], addressing the behavior of the Oracle is much more complex than applying a simple neighborhood approach.

On the other hand, DES techniques based on other criteria, such as the degree of consensus of the ensemble classifiers [15; 16], encounter some problems when the search cannot find a consensus among the ensembles. In addition, they neglect the local performance of the base classifiers. As stated by the "No Free Lunch" theorem [33], no algorithm is better than any other over all possible classes of problems. Using a single criterion to measure the level of competence of a base classifier is very error-prone. Thus, we believe that multiple criteria to measure the competence of a base classifier should be taken into account in order to achieve a more robust dynamic ensemble selection technique.

In this paper, we propose a novel dynamic ensemble selection framework using meta-learning. From the meta-learning perspective, the dynamic ensemble selection problem is considered as another classification problem, called meta-problem. The meta-features of the meta-problem are the different criteria used to measure the level of competence of the base classifier. We propose five sets of meta-features in this paper. Each set captures a different property about the behavior of the base classifier, and can be seen as a different dynamic selection criterion such as, the classification performance in a local region of the feature space and the classifier confidence for the classification of the input sample. Using five distinct sets of meta-features, even though one criterion might fail due to problems in the local regions of the feature space [20] or due to low confidence results [35], the system can still achieve a good performance as other meta-features are also considered by the selection scheme. Furthermore, in a recent analysis [55] we compared the criteria used to measure the competence of base classifiers embedded in different DES techniques. The result demonstrates that, given the same query sample, distinct DES criteria select a different base classifier as the most competent one. Thus, they are not fully correlated. Hence, we believe that a more robust dynamic ensemble selection technique is achieved using five sets of meta-features rather than only one.

The meta-features are used as input to a meta-classifier that decides whether or not a base classifier is competent enough for the classification of an input sample based on the meta-features. The use of meta-learning has recently been proposed in [56] as an alternative for performing classifier selection in static scenarios. We believe that we can carry this further, and extend the use of meta-learning to dynamically estimate the level of competence of a base classifier.

The proposed framework is divided into three phases: overproduction, meta-training and generalization. In the overproduction stage, a pool of classifiers is generated using the training data. In the meta-training stage, the five sets of meta-features are extracted from the training data, and are used to train the meta-classifier that works as the classifier selector. During the generalization phase, the meta-features are extracted from the query instance and passed down as inputs to the meta-classifier. The meta-classifier estimates whether a base classifier is competent enough to classify the given test instance. Thus, the proposed system differs from the current state-of-the-art dynamic selection techniques not only because it uses multiple criteria to perform the classifier selection, but also because the classifier selection rule is learned by the meta-classifier using the training data.

The generalization performance of the system is evaluated over 30 classification problems. We compare the proposed framework against eight state-of-the-art dynamic selection techniques as well as static combination methods. The evaluation is focused on small size dataset, since DES techniques has shown to be an effective tool for problems where the level of uncertainty for recognition is high due to few training samples [16]. However, a few larger datasets were also considered in order to evaluate the performance of the proposed framework under different conditions. The goal of the experiments is to answer the following research questions: (1) Can the use of multiple DES criteria, as meta-features, lead to a more robust dynamic selection technique? (2) Does the proposed framework outperform current DES techniques for ill-defined problems?

This paper is organized as follows: Section 2.2 introduces the notion of classifier competence, and the state-of-the-art techniques for dynamically measuring the classifiers' competence are presented. The proposed framework is presented in Section 2.3. The experimental study is conducted in Section 2.4. Finally, our conclusion is presented in the last section.

2.2 Classifier competence for dynamic selection

Classifier competence defines how much we trust an expert, given a classification task. The notion of competence used is extensively in the field of machine learning as a way of selecting, from the plethora of different classification models, the one that best fits the given problem. Let $C = \{c_1, \ldots, c_M\}$ (M is the size of the pool of classifiers) be the pool of classifiers and c_i a base classifier belonging to the pool C. The goal of dynamic selection is to find an ensemble of classifiers $C' \subset C$ that has the best classifiers to classify a given test sample $\mathbf{x}_j$. This is different from static selection, where the ensemble of classifiers C' is selected during the training phase, and considering the global performance of the base classifiers over a validation dataset [10; 11; 12; 13].

Nevertheless, the key issue in dynamic selection is how to measure the competence of a base classifier c_i for the classification of a given query sample $\mathbf{x}_j$. In the literature, we can observe three categories: the classifier accuracy over a local region, i.e., in a region of the feature space surrounding the query instance $\mathbf{x}_j$, decision templates [57], which are techniques that work in the decision space (i.e, a space defined by the outputs of the base classifiers) and the extent of consensus or confidence. The three categories are described in the following subsections.

2.2.1 Classifier accuracy over a local region

Classifier accuracy is the most commonly used criterion for dynamic classifier and ensemble selection techniques [22; 14; 20; 30; 23; 28; 38; 29; 19]. Techniques that are based on local accuracy first define a small region in the feature space surrounding a given test instance $\mathbf{x}_j$, called the region of competence. This region is computed using either the K-NN algorithm [14;

22; 20] or by Clustering techniques [30; 23], and can be defined either in the training set [22] or in the validation set, such as in the KNORA techniques [14].

Based on the samples belonging to the region of competence, a criterion is applied in order to measure the level of competence of a base classifier. For example, the Overall Local Accuracy (OLA) [22] technique uses the accuracy of the base classifier in the whole region of competence as a criterion to measure its level of competence. The classifier that obtains the highest accuracy rate is considered the most competent one. The Local Classifier Accuracy (LCA) [22] computes the performance of the base classifier in relation to a specific class label using a posteriori information [44]. The Modified Local Accuracy [29] works similarly to the LCA technique, with the only difference being that each sample belonging to the region of competence is weighted by its Euclidean distance to the query instance. That way, instances from the region of competence that are closer to the test sample have a higher influence when computing the performance of the base classifier. The classifier rank method [38] uses the number of consecutive correctly classified samples as a criterion to measure the level of competence. The classifier that correctly classifies the most consecutive samples coming from the region of competence is considered to have the highest competence level or "rank".

Ko et al. [14] proposed the K-Nearest Oracles (KNORA) family of techniques, inspired by the Oracle concept. Four techniques are proposed: the KNORA-Eliminate (KNORA-E) which, considers that a base classifier c_i is competent for the classification of the query instance $\mathbf{x}_j$ if c_i achieves a perfect accuracy for the whole region of competence. Only the base classifiers with a perfect accuracy are used during the voting scheme. In The KNORA-Union (KNORA-U) technique, the level of competence of a base classifier c_i is measured by the number of correctly classified samples in the defined region of competence. In this case, every classifier that correctly classified at least one sample can submit a vote. In addition, two weighted versions, KNORA-E-W and KNORA-U-W were also proposed, in which the influence of each sample belonging to the region of competence was weighted based on its Euclidean distance to the query sample $\mathbf{x}_j$. Lastly, Xiao et al. [19] proposed the Dynamic Classifier Ensemble for Imbalanced Data (DCEID), which is based on the same principles as the LCA technique.

However, this technique also takes into account each class prior probability when computing the performance of the base classifier for the defined region of competence in order to deal with imbalanced distributions.

The difference between these techniques lies in how they utilize the local accuracy information in order to measure the level of competence of a base classifier. The main issue with the techniques arises from the fact that they depend on the performance of the techniques that define the region of competence such as K-NN or clustering techniques. In our previous work [20], we demonstrated that the effectiveness of dynamic selection techniques is limited by the performance of the algorithm that defines the region of competence. The dynamic selection technique is likely to commit errors when outlier instances (i.e., mislabelled samples) exists around the query sample in the feature space [20]. Using the local accuracy information alone is not sufficient to achieve results close to the Oracle. Moreover, any difference between the distribution of validation and test datasets may negatively affect the system performance. Consequently, we believe that additional information should also be considered.

2.2.2 Decision Templates

In this class of methods, the goal is also to select samples that are close to the query instance $\mathbf{x}_j$. However, the similarity is computed over the decision space through the concept of decision templates [57]. This is performed by transforming both the test instance $\mathbf{x}_j$ and the validation data into output profiles. The output profile of an instance $\mathbf{x}_j$ is denoted by $\tilde{\mathbf{x}}_j = \left\{\tilde{\mathbf{x}}_{j,1}, \tilde{\mathbf{x}}_{j,2}, \ldots, \tilde{\mathbf{x}}_{j,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,i}$ is the decision yielded by the base classifier c_i for the sample $\mathbf{x}_j$.

Based on the information extracted from the decision space, the K-Nearest Output Profile (KNOP) [17] is similar to the KNORA technique, with the difference being that the KNORA works in the feature space, while the KNOP works in the decision space. The KNOP technique first defines a set with the samples that are most similar to the output profile of the input sample, $\tilde{\mathbf{x}}_j$ in the decision space, called the output profiles set. The validation set is used for this

purpose. Then, similarly to the KNORA-E technique, only the base classifiers that achieve a perfect recognition accuracy for the samples belonging to the output profiles set are used during the voting scheme. The Multiple Classifier Behaviour (MCB) technique [21] also defines a set with the most similar output profiles to the input sample using the decision space. Here, the selection criterion is based on a threshold. The base classifiers that achieve a performance higher than the predefined threshold are considered competent and are selected to form the ensemble.

The advantage of this class of methods is that they are not limited by the quality of the region of competence defined in the feature space, with the similarity computed based on the decision space rather than the feature space. However, the disadvantage with this comes from the fact that only global information is considered, while the local expertise of each base classifier is neglected.

2.2.3 Extent of Consensus or confidence

Different from other methods, techniques that are based on the extent of consensus work by considering a pool of ensemble of classifiers (EoC) rather than a pool of classifiers. Hence, the first step is to generate a population of EoC, $C^* = \{C'_1, C'_2, \ldots, C'_{M'}\}$ (M' is the number of EoC generated) using an optimization algorithm such as genetic algorithms or greedy search [12; 15; 13]. Then, for each new query instance $\mathbf{x}_j$, the level of competence of an ensemble of classifiers C'_i is equal to the extent of consensus among its base classifiers.

Several criterion based on this paradigm was proposed: the Margin-based Dynamic Selection (MDS) [15], where the criterion is the margin between the most voted class and the second most voted class. The margin is computed simply by considering the difference between the number of votes received by the most voted class and those received by the second most voted class. Two variations of the MDS where proposed in [15], the Class-Strength Dynamic Selection (CSDS), which includes the ensemble decision in the computation of the MDS, and the GSDS, where the global performance of each EoC is also taken into account [16]. Another technique

from this paradigm is the Ambiguity-guided Dynamic Selection (ADS) [15], which uses the ambiguity among the base classifiers of an EoC as the criterion for measuring the competence level of an EoC. The ambiguity is calculated by the number of base classifiers of an ensemble that disagrees with the ensemble decision. The lower the number of classifiers that disagree with the ensemble decision, the higher the level of competence of the EoC.

The greatest advantage of this class of methods stems from the fact that it does not require information from the region of competence. Thus, it does not suffer from the limitations of the algorithm that defines the region of competence. However, these techniques present the following disadvantages: In many cases, the search cannot find an EoC with an acceptable confidence level. There is a tie between different members of the pool, and the systems end up performing a random decision [16]. In addition, some classifiers are more overtrained than others. In this case, they end up dominating the outcome even though they do not present better recognition performance [7]. The pre-computation of ensembles also greatly increases the overall system complexity as we are dealing with a pool of EoC rather than a pool of classifiers.

2.3 The Proposed Framework: META-DES

2.3.1 Problem definition

From the meta-learning perspective, the dynamic selection problem can be seen as another classification problem, called the meta-problem. This meta-problem uses different criteria regarding the behavior of a base classifier in order to decide whether it is competent enough to classify a given sample $\mathbf{x}_j$. Thus, a dynamic selection system can be defined based on two environments. A classification environment in which the input features are mapped into a set of class labels $w = \{w_1, w_2, ..., w_L\}$ and a meta-classification environment in which information about the behavior of the base classifier is extracted from the classification environment and used to decide whether a base classifier c_i is competent enough to classify $\mathbf{x}_j$.

To keep with the conventions of the meta-learning literature, we define the proposed dynamic ensemble selection in a meta-learning framework as follows:

- The **meta-problem** consists in defining whether a base classifier c_i is competent enough to classify $\mathbf{x}_j$.
- The **meta-classes** of this meta-problem are either "competent" or "incompetent" to classify $\mathbf{x}_j$.
- Each **meta-feature** f_i corresponds to a different criterion to measure the level of competence of a base classifier.
- The meta-features are encoded into a **meta-features vector** $v_{i,j}$ which contains the information about the behavior of a base classifier c_i in relation to the input instance $\mathbf{x}_j$.
- A **meta-classifier** λ is trained based on the meta-features $v_{i,j}$ to predict whether or not c_i will achieve the correct prediction for $\mathbf{x}_j$.

Thus, the proposed system differs from the current state-of-the-art dynamic selection techniques not only because it uses multiple criteria, but also because the selection rule is learned by the meta-classifier λ using the training data.

2.3.2 The proposed META-DES

The META-DES framework is divided into three phases (Figure 2.1):

a. The overproduction phase, where the pool of classifiers $C = \{c_1, \ldots, c_M\}$, composed of M classifiers, is generated using the training instances $\mathbf{x}_{j,train}$ from the dataset $\mathcal{T}$.

b. The meta-training stage, in which samples $\mathbf{x}_{j,train_\lambda}$ from the meta-training dataset $\mathcal{T}_\lambda$ are used to extract the meta-features. A different dataset $\mathcal{T}_\lambda$ is used in this phase in order to prevent overfitting. The meta-feature vectors $v_{i,j}$ are stored in the set $\mathcal{T}_\lambda^*$ that is later used to train the meta-classifier λ.

c. The generalization phase, given a test sample $\mathbf{x}_{j,test}$ resulting from the generalization data $\mathcal{G}$; its region of competence is extracted using the samples from the dynamic selection dataset D_{SEL} in order to compute the meta-features. The meta-feature vector $v_{i,j}$ is then passed to the selector λ, which decides whether c_i is competent enough to classify $\mathbf{x}_{j,test}$ and should be added to the ensemble, C'. The majority vote rule is applied over the ensemble C', giving the classification w_l of $\mathbf{x}_{j,test}$.

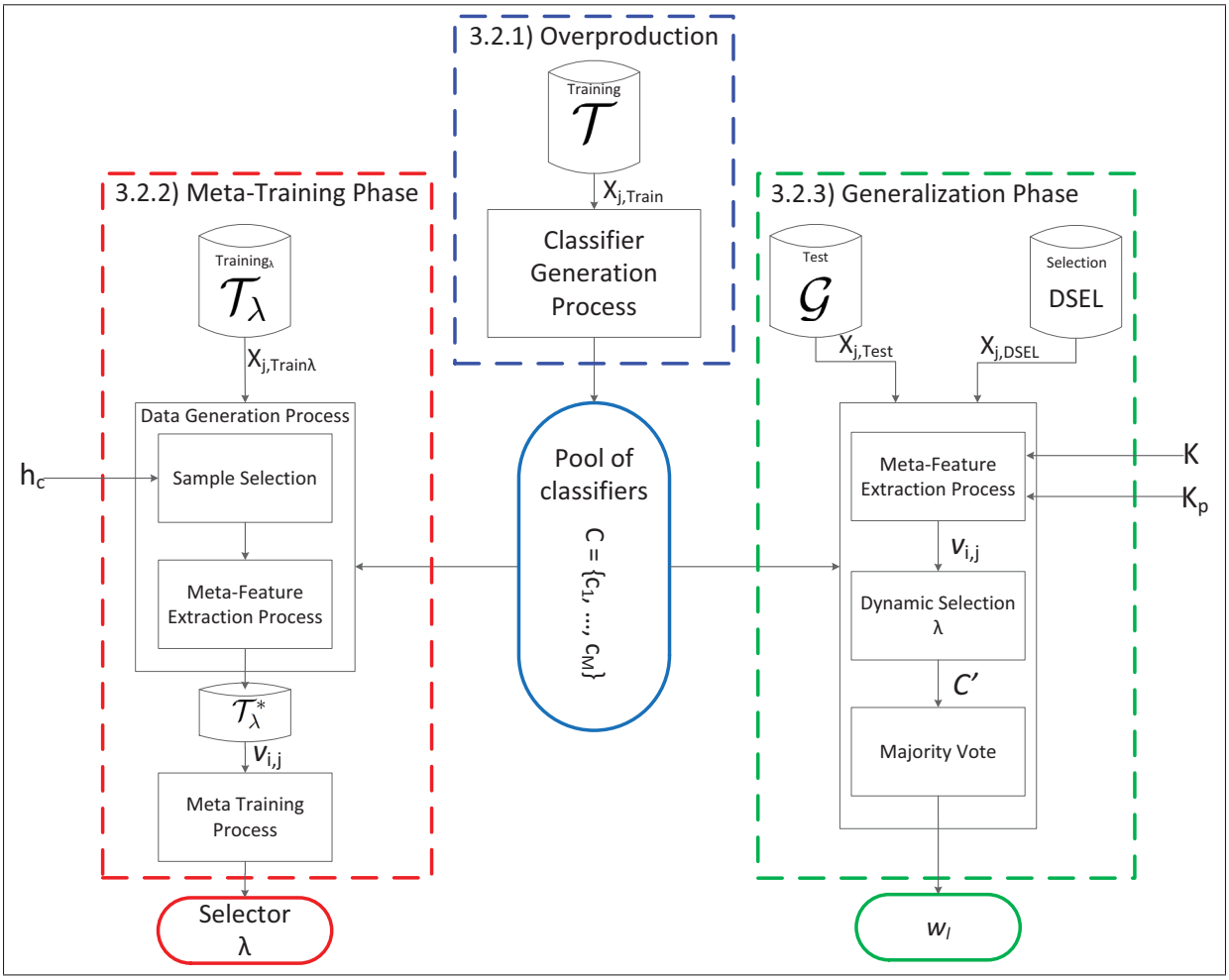


Figure 2.1 Overview of the proposed META-DES framework. It is divided into three steps 1) Overproduction, where the pool of classifiers $C = \{c_1, \ldots, c_M\}$ is generated, 2) The training of the meta-classifier λ, and 3) The generalization phase where an ensemble C' is dynamically defined based on the meta-information extracted from $\mathbf{x}_{j,test}$ and the pool $C = \{c_1, \ldots, c_M\}$. The generalization phase returns the label w_l of $\mathbf{x}_{j,test}$. h_C, K and K_p are the hyper-parameters required by the proposed system

2.3.2.1 Overproduction

In this work, the Overproduction phase is performed using the Bagging technique [3; 58]. Bagging is an acronym for Bootstrap AGGregatING. The idea behind this technique is to build a diverse ensemble of classifiers by randomly selecting different subsets of the training data. Each subset is used to train one individual classifier c_i. As the focus of the paper is on classifier selection, and not on classifier generation methods, only the bagging technique is considered.

2.3.2.2 Meta-training

As shown in Figure 2.1, the meta-training stage consists of three steps: the sample selection process, the meta-features extraction process, and the training of the meta-classifier λ. For every sample $\mathbf{x}_{j,train_\lambda} \in \mathcal{T}_\lambda$, the first step is to apply the sample selection mechanism in order to know whether or not $\mathbf{x}_{j,train_\lambda}$ should be used for the training of the meta-classifier λ. The whole Meta-training phase is formalized in Algorithm 2.1.

2.3.2.2.1 Sample Selection

As demonstrated by Dos Santos et al. [15] and Cavalin et al. [16], one of the main issues in dynamic ensemble selection arises when classifying testing instances where the degree of consensus among the pool of classifier is low, i.e., when the number of votes from the winning class is close or even equal to the number of votes from the second class. To tackle this issue, we decided to focus the training of the meta-classifier λ to specifically deal with cases where the extent of consensus among the pool is low. This step is conducted using a threshold h_C, called the consensus threshold. Each instance $\mathbf{x}_{j,train_\lambda}$ is first evaluated by the whole pool of classifiers in order to compute the degree of consensus among the pool, denoted by $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$. If the consensus $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$ falls below the consensus threshold h_C, the instance $\mathbf{x}_{j,train_\lambda}$ is used to compute the meta-features.

Input: Training data $\mathcal{T}_\lambda$
Input: Pool of classifiers $C = \{c_1, \ldots, c_M\}$
1: $\mathcal{T}_\lambda^* = \emptyset$
2: **for all** $\mathbf{x}_{j,train_\lambda} \in \mathcal{T}_\lambda$ **do**
3: Compute the consensus of the pool $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$
4: **if** $H\left(\mathbf{x}_{j,train_\lambda}, C\right) < h_C$ **then**
5: Find the region of competence θ_j of $\mathbf{x}_{j,train_\lambda}$ using $\mathcal{T}_\lambda$.
6: Compute the output profile $\tilde{\mathbf{x}}_{j,train_\lambda}$ of $\mathbf{x}_{j,train_\lambda}$.
7: Find the K_p similar output profiles ϕ_j of $\tilde{\mathbf{x}}_{j,train_\lambda}$ using $\tilde{\mathcal{T}_\lambda}$.
8: **for all** $c_i \in C$ **do**
9: $v_{i,j} = MetaFeatureExtraction(\theta_j, \phi_j, c_i, \mathbf{x}_{j,train_\lambda})$
10: **if** c_i correctly classifies $\mathbf{x}_{j,train_\lambda}$ **then**
11: $\alpha_{i,j} = 1$ "c_i is competent for $\mathbf{x}_{j,train_\lambda}$"
12: **else**
13: $\alpha_{i,j} = 0$ " c_i is incompetent for $\mathbf{x}_{j,train_\lambda}$"
14: **end if**
15: $\mathcal{T}_\lambda^* = \mathcal{T}_\lambda^* \cup \left\{v_{i,j}\right\}$
16: **end for**
17: **end if**
18: **end for**
19: Divide $\mathcal{T}_\lambda^*$ into 25% for validation and 75% for training.
20: Train λ using the Levenberg-Marquadt algorithm.
21: **return** The meta-classifier λ.

Algorithm 2.1: The Meta-Training Phase

Before extracting the meta-features, the region of competence of the instance $\mathbf{x}_{j,train_\lambda}$, denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$, must first be computed. The region of competence θ_j is defined in the $\mathcal{T}_\lambda$ set, using the K-Nearest Neighbor algorithm (line 5). Then, $\mathbf{x}_{j,train_\lambda}$ is transformed into an output profile. The output profile of the instance $\mathbf{x}_{j,train_\lambda}$ is denoted by $\tilde{\mathbf{x}}_{j,train_\lambda} = \left\{\tilde{\mathbf{x}}_{j,train_\lambda,1}, \tilde{\mathbf{x}}_{j,train_\lambda,2}, \ldots, \tilde{\mathbf{x}}_{j,train_\lambda,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,train_\lambda,i}$ is the decision yielded by the base classifier c_i for the sample $\mathbf{x}_{j,train_\lambda}$ [16; 17].

Next, with the region of competence θ_j and the set with the most similar output profiles ϕ_j computed, for each base classifier c_i belonging to the pool of classifiers C, one meta-feature vector $v_{i,j}$ is extracted (lines 8 to 14). Each $v_{i,j}$ contains five sets of meta-features:

2.3.2.2.2 Meta-feature extraction process

Five different sets of meta-features are proposed in this work. Each feature set f_i, corresponds to a different criterion for measuring the level of competence of a base classifier. Each set captures a different property about the behavior of the base classifier, and can be seen as a different criterion to dynamically estimate the level of competence of base classifier such as, the classification performance estimated in a local region of the feature space and the classifier confidence for the classification of the input sample. Using five distinct sets of meta-features, even though one criterion might fail due to imprecisions in the local regions of the feature space or due to low confidence results, the system can still achieve a good performance as other meta-features are considered by the selection scheme. Table 2.1 shows the criterion used by each f_i and its relationship with one dynamic ensemble selection paradigm presented in Section 2.2.

Table 2.1 Relationship between each meta-features and different paradigms to compute the level of competence of a base classifier

Meta-Feature	**Criterion**	**Paradigm**
f_1	Local accuracy in the region of competence	Classifier Accuracy over a local region
f_2	Extent of consensus in the region of competence	Classifier consensus
f_3	Overall accuracy in the region of competence	Accuracy over a local region
f_4	Accuracy in the decision space	Decision Templates
f_5	Degree of confidence for the input sample	Classifier confidence

Three meta-features, f_1, f_2 and f_3, are computed using information extracted from the region of competence θ_j. f_4 uses information extracted from the set of output profiles ϕ_j. f_5 is calculated directly from the input sample $\mathbf{x}_{j,train_\lambda}$, and corresponds to the level of confidence of c_i for the classification of $\mathbf{x}_{j,train_\lambda}$.

f_1 **- Neighbors' hard classification:** First, a vector with K elements is created. For each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of the vector is set to 1, otherwise it is 0. Thus, K meta-features are computed.

f_2 **- Posterior probability:** First, a vector with K elements is created. Then, for each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, the posterior probability of c_i, $P(w_l \mid \mathbf{x}_k)$ is computed and inserted into the k-th position of the vector. Consequently, K meta-features are computed.

f_3 **- Overall Local accuracy:** The accuracy of c_i over the whole region of competence θ_j is computed and encoded as f_3.

f_4 **- Output profiles classification:** First, a vector with K_p elements is generated. Then, for each member $\tilde{\mathbf{x}}_k$ belonging to the set of output profiles ϕ_j, if the label produced by c_i for $\mathbf{x}_k$ is equal to the label $w_{l,k}$ of $\tilde{\mathbf{x}}_k$, the k-th position of the vector is set to 1, otherwise it is 0. A total of K_p meta-features are extracted using output profiles.

f_5 **- Classifier's Confidence:** The perpendicular distance between the input sample $\mathbf{x}_{j,train_\lambda}$ and the decision boundary of the base classifier c_i is calculated and encoded as f_5. f_5 is normalized to a $[0-1]$ range using the Min-max normalization.

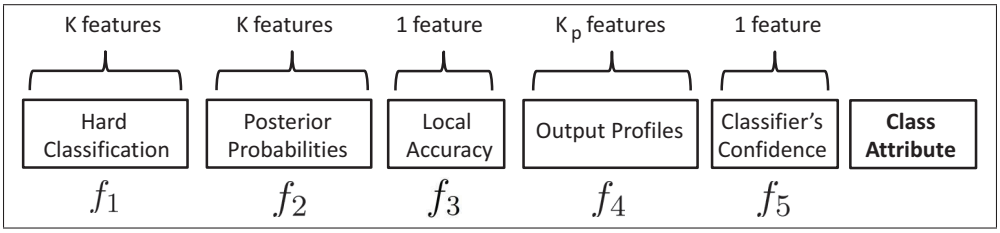


Figure 2.2 Feature Vector containing the meta-information about the behavior of a base classifier. A total of 5 different meta-features are considered. The size of the feature vector is $(2 \times K) + K_p + 2$. The class attribute indicates whether or not c_i correctly classified the input sample

A vector $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ is obtained at the end of the process (Figure 2.2). If c_i correctly classifies $\mathbf{x}_{j,train_\lambda}$, the class attribute of $v_{i,j}$, $\alpha_{i,j} = 1$ (i.e., $v_{i,j}$ corresponds to the behavior of a competent classifier), otherwise $\alpha_{i,j} = 0$. $v_{i,j}$ is stored in the meta-features dataset $\mathcal{T}_\lambda^*$ (lines 10 to 16).

For each sample $\mathbf{x}_{j,train_\lambda}$ used in the meta-training stage, a total of M (M is the size of the pool of classifiers C) meta-feature vectors $v_{i,j}$ are extracted, each one corresponding to one classifier from the pool C. In this way, the size of the meta-training dataset $\mathcal{T}_\lambda^*$ is the pool size $M\times$ number of training samples N. For instance, consider that 200 training samples are available for the meta-training stage ($N = 200$), if the pool C is composed of 100 weak classifiers ($M = 100$), the meta-training dataset is the number of training samples $N \times$ the number classifiers in the pool M, $N * M = 20.000$. Hence, even though the classification problem may be ill-defined due to the size of the training set, we can overcome this limitation in the meta-problem by increasing the size of the pool of classifiers.

2.3.2.2.3 Training

The last step of the meta-training phase is the training of the meta-classifier λ. The dataset $\mathcal{T}_\lambda^*$ is divided on the basis of 75% for training and 25% for validation. A Multi-Layer Perceptron (MLP) neural network is considered as the selector λ. The validation data was used to select the number of nodes in the hidden layer. We use a configuration of 10 neurons in the hidden layer since there were no improvement in results with more than 10 neurons. The training process for λ is performed using the Levenberg-Marquadt algorithm. In addition, the training process is stopped if its performance on the validation set decreases or fails to improve for five consecutive epochs.

2.3.2.3 Generalization Phase

The generalization procedure is formalized by Algorithm 2.2. Given the query sample $\mathbf{x}_{j,test}$, in this phase, the region of competence θ_j is computed using the samples from the dynamic selection dataset D_{SEL} (line 2). Following that, the output profiles $\tilde{\mathbf{x}}_{j,test}$ of the test sample, $\mathbf{x}_{j,test}$, are calculated. The set with K_p similar output profiles ϕ_j, of the query sample $\mathbf{x}_{j,test}$, is obtained through the Euclidean distance applied over the output profiles of the dynamic selection dataset, $\tilde{D}_{SEL}$.

Input: Query sample $\mathbf{x}_{j,test}$
Input: Pool of classifiers $C = \{c_1, \ldots, c_M\}$
Input: dynamic selection dataset D_{SEL}
1: $C' = \emptyset$
2: Find the region of competence θ_j of $\mathbf{x}_{j,test}$ using D_{SEL}.
3: Compute the output profile $\tilde{\mathbf{x}}_{j,test}$ of $\mathbf{x}_{j,test}$.
4: Find the K_p similar output profiles ϕ_j of $\tilde{\mathbf{x}}_{j,test}$ using $\tilde{D}_{SEL}$.
5: **for all** $\mathbf{c}_i \in C$ **do**
6: $v_{i,j} = FeatureExtraction(\theta_j, \phi_j, c_i, \mathbf{x}_{j,test})$
7: input $v_{i,j}$ to λ
8: **if** $\alpha_{i,j} = 1$ "c_i is competent for $\mathbf{x}_{j,test}$" **then**
9: $C' = C' \cup \{c_i\}$
10: **end if**
11: **end for**
12: $w_l = MajorityVote(\mathbf{x}_{j,test}, C')$
13: **return** w_l

Algorithm 2.2: Classification steps using the selector λ

Next, for each classifier c_i belonging to the pool of classifiers C, the meta-feature extraction process is called (Section 3.2.2.2), returning the meta-features vector $v_{i,j}$ (lines 5 and 6). Then, $v_{i,j}$ is used as input to the meta-classifier λ. If the output of λ is 1 (i.e., competent), c_i is included in the ensemble C' (lines 8 to 10). After every base classifier, c_i, is evaluated, the ensemble C' is obtained. The base classifiers in C' are combined through the Majority Vote rule [24], giving the label w_l of $x_{j,test}$ (line 12 and 13). The majority vote rule is used to combine the selected classifiers since it has been successfully used by other DES techniques [1]. Tie-breaking is handled by choosing the class with the highest a posteriori probability.

2.4 Experiments

2.4.1 Datasets

A total of 30 datasets are used in the comparative experiments. sixteen coming from the UCI machine learning repository [59], four from the STATLOG project [60], four from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [61], four from the Lud-

mila Kuncheva Collection of real medical data [62], and two artificial datasets generated with the Matlab PRTOOLS toolbox [63]. We consider both ill-defined problems, such as, Heart and Liver Disorders as well as larger databases, such as, Adult, Magic Gamma Telescope, Phoneme and WDG V1. The key features of each dataset are shown in Table 2.2.

Table 2.2 Key Features of the datasets used in the experiments

Database	No. of Instances	Dimensionality	No. of Classes	Source
Pima	768	8	2	UCI
Liver Disorders	345	6	2	UCI
Breast (WDBC)	568	30	2	UCI
Blood transfusion	748	4	2	UCI
Banana	1000	2	2	PRTOOLS
Vehicle	846	18	4	STATLOG
Lithuanian	1000	2	2	PRTOOLS
Sonar	208	60	2	UCI
Ionosphere	315	34	2	UCI
Wine	178	13	3	UCI
Haberman's Survival	306	3	2	UCI
Cardiotocography (CTG)	2126	21	3	UCI
Vertebral Column	310	6	2	UCI
Steel Plate Faults	1941	27	7	UCI
WDG V1	50000	21	3	UCI
Ecoli	336	7	8	UCI
Glass	214	9	6	UCI
ILPD	214	9	6	UCI
Adult	48842	14	2	UCI
Weaning	302	17	2	LKC
Laryngeal1	213	16	2	LKC
Laryngeal3	353	16	3	LKC
Thyroid	215	5	3	LKC
German credit	1000	20	2	STATLOG
Heart	270	13	2	STATLOG
Satimage	6435	19	7	STATLOG
Phoneme	5404	6	2	ELENA
Monk2	4322	6	2	KEEL
Mammographic	961	5	2	KEEL
MAGIC Gamma Telescope	19020	10	2	KEEL

2.4.2 Experimental Protocol

The experiments were conducted using 20 replications. For each replication, the datasets were randomly divided on the basis 50% for training, 25% for the dynamic selection dataset (D_{SEL}), and 25% for the test set ($\mathcal{G}$). The divisions were performed maintaining the priors probabilities of each class. For the proposed META-DES, 50% of the training data was used in the meta-training process $\mathcal{T}_\lambda$ and 50% for the generation of the pool of classifiers ($\mathcal{T}$).

For the two-class classification problems, the pool of classifiers was composed of 100 Perceptrons generated using the bagging technique [3]. For the multi-class problems, the pool of classifiers was composed of 100 multi-class perceptron classifier. The use of Perceptron as base classifier comes from the following observations based on past works in the literature:

- The use of weak classifiers can show more differences between the DES schemes [14]. Thus, making it a better option for comparing different DES techniques.
- Past works in the DES literature demonstrate that the use of weak models as base classifier achieve better results [15; 16; 64; 65; 20], where the use of decision trees or Perceptrons outperform strong classification models such as KNN classifiers.
- As reported by Leo Breiman [3; 58], the bagging technique achieves better results when weak and unstable base classifiers are used.

2.4.3 Parameters Setting

The performance of the proposed selection scheme depends on three parameters: the neighborhood size, K, the number of similar patterns using output profiles K_p and the consensus threshold h_C. The dynamic selection dataset D_{SEL} was used for the analysis. The following methodology is used:

- For the sake of simplicity, we selected the parameters that performed best.
- The value of the parameter K was selected based on the results of our previous paper [20]. In this case, $K = 7$ showed the best overall results, considering several dynamic selection techniques.
- The Kruskall-Wallis statistical test with a 95% confidence interval was used to determine whether the difference in results was statistically significant. If two configurations yielded similar results, we selected the one with the smaller parameter value as it leads to a smaller meta-features vector.

- The parameter h_C was evaluated with K_p initially set at 1.
- The best value of h_c was used in the evaluation of the best value for K_p.
- Only a subset with eleven of the thirty datasets are used for parameters setting procedure: Pima, Liver, Breast, Blood Transfusion, Banana, Vehicle, Lithuanian, Sonar, Ionosphere, Wine, Haberman's Survival.

2.4.3.1 The effect of the parameter h_C

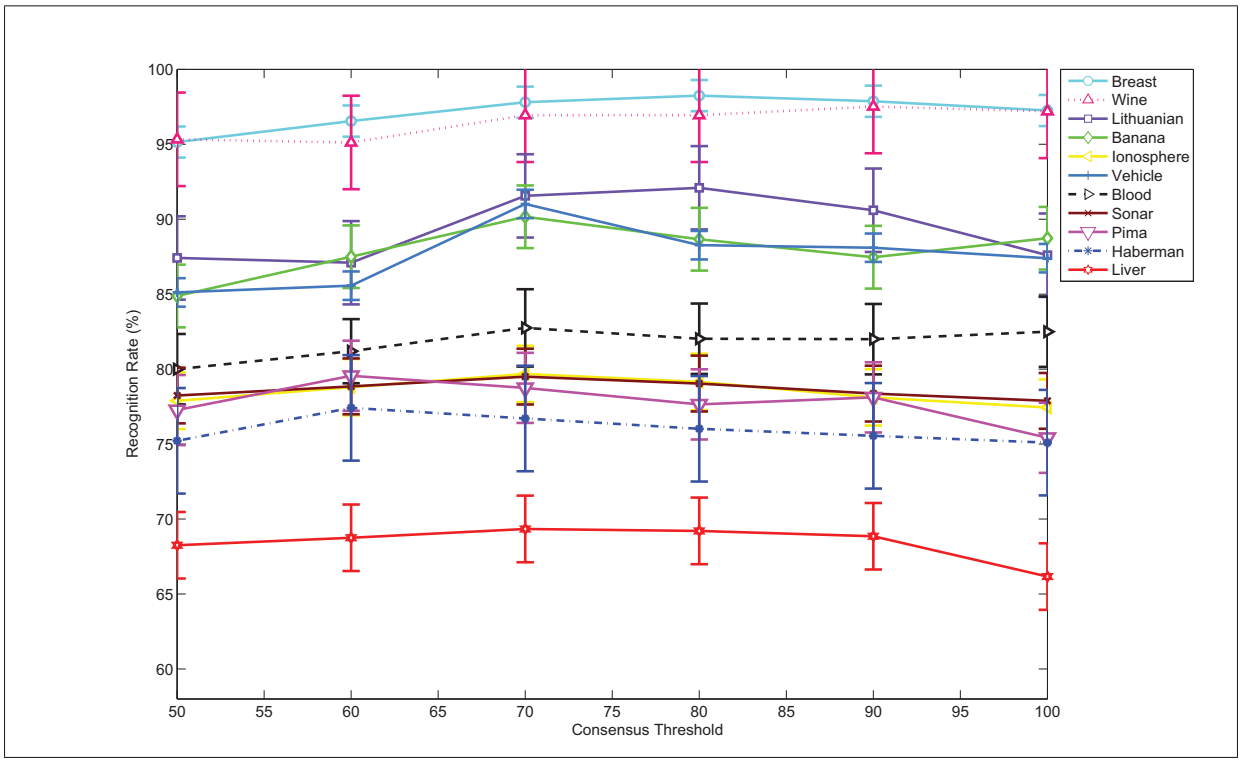


Figure 2.3 Performance of the proposed system based on the parameter h_C on the dynamic selection dataset, D_{SEL}. $K = 7$ and $K_p = 1$

We varied the parameter h_c from 50% to 100% at 10 percentile point interval. Figure 2.3 shows the mean performance and standard deviation for each h_C value. We compared each pair of results using the Kruskal-Wallis non-parametric statistical test with a 95% confidence interval. For 6 out of 11 datasets (Vehicle, Lithuanian, Banana, Blood transfusion, Ionosphere

and Sonar) $h_C = 70\%$ presented a value that was statistically superior to the others. Hence, $h_C = 70\%$ was selected.

2.4.3.2 The effect of the parameter K_p

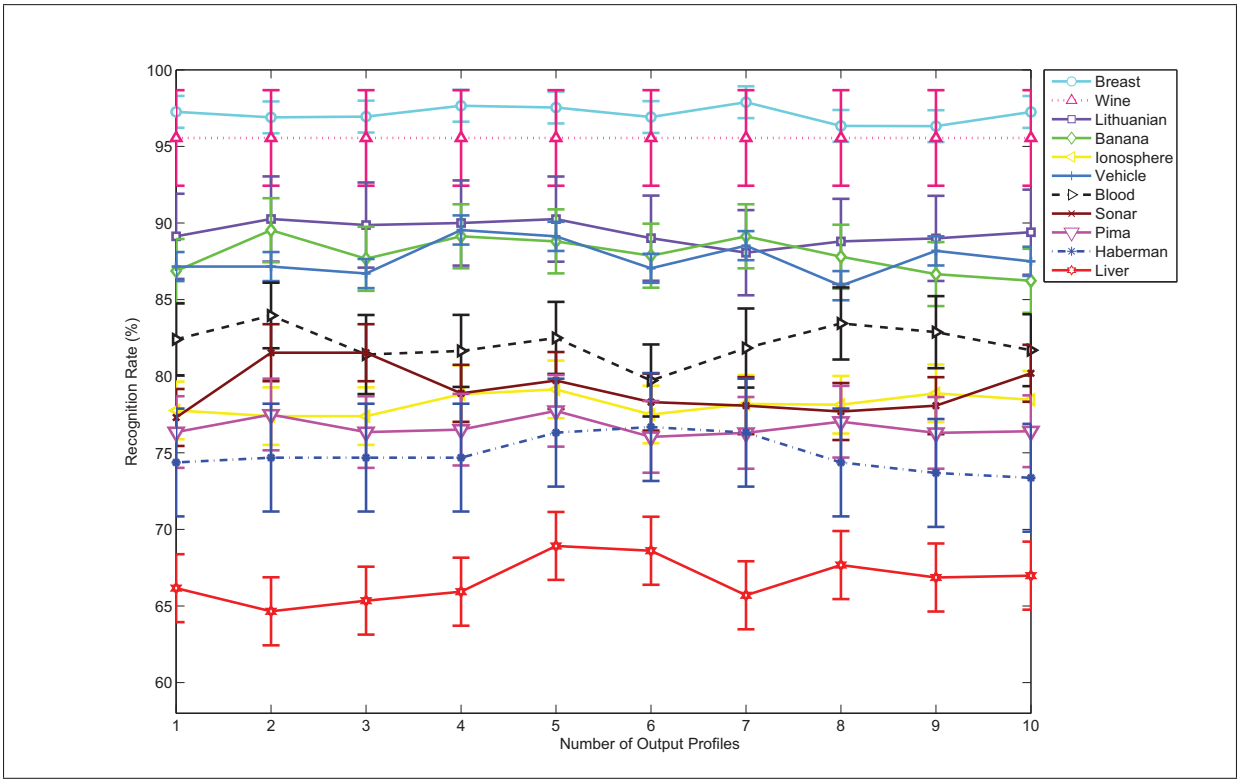


Figure 2.4 The performance of the system varying the parameter K_p from 1 to 10 on the dynamic selection dataset, D_{SEL}. $h_c = 70\%$ and $K = 7$

Figure 2.4 shows the impact of the value of the parameter K_p in an 1-to-10 range. Once again, we compared each pair of results using the Kruskal-Wallis non-parametric statistical test, with a 95% confidence. The results were statistically different only for the Sonar, Ionosphere and liver disorders datasets, where the value of $K_p = 5$ showed the best results. Hence, K_p was set at 5.

2.4.4 Comparison with the state-of-the-art dynamic selection techniques

In this section we compare the recognition rates obtained by the proposed META-DES, against eight dynamic selection techniques found in the literature [1]. The objective of this comparative study is to answer the following research question: (1) Can the use of multiple DES criteria as meta-features lead to a more robust dynamic selection technique? (2) Does the proposed framework outperform current DES techniques for ill-defined problems?

The eight state-of-the-art DES techniques used in this study are: the KNORA-ELIMINATE [14], KNORA-UNION [14], DES-FA [20], Local Classifier Accuracy (LCA) [22], Overall Local Accuracy (OLA) [22], Modified Local Accuracy (MLA) [29], Multiple Classifier Behaviour (MCB) [21] and K-Nearests Output Profiles (KNOP) [17; 16]. These techniques were selected because they presented the very best results in the dynamic selection literature according to a recent survey on this topic [1]. In addition, we also compare the performance of the proposed META-DES with static combination methods (Adaboost and Bagging), the classifier with the highest accuracy in the validation data (Single Best), static ensemble selection based on the majority voting error [66] and the abstract model (Oracle) [32]. The Oracle represents the ideal classifier selection scheme. It always selects the classifier that predicted the correct label, for any given query sample, if such classifier exists. For the static ensemble selection method, 50% of the classifiers of the pool are selected. The comparison against static methods is used since it is suggested the DES literature that the minimum requirement for a DES method is to surpass the performance of static selection and combination methods in the same pool [1].

For all techniques, the pool of classifiers C is composed of 100 Perceptrons as base classifier ($M = 100$). For the state-of-the-art DES techniques (KNORA-E, KNORA-U, DES-FA, LCA, OLA, MLA, MCB and KNOP), the size of the region of competence (neighborhood size), K is set to 7, since it achieved the best result on previous publications [1; 20]. The size of the region of competence K is the only hyper-parameter required for the eight DES techniques. For the Adaboost and Bagging technique 100 iterations are used (i.e., 100 base classifier are generated).

Table 2.3 Mean and standard deviation results of the accuracy obtained for the proposed DES and the DES systems in the literature. A pool of 100 Perceptrons as base classifiers is used for all techniques. The best results are in bold. Results that are significantly better ($p < 0.05$) are marked with a •

Database	META-DES	KNORA-E	KNORA-U	DES-FA	LCA	OLA	MLA	MCB	KNOP
Pima	**79.03(2.24)** •	73.79(1.86)	76.60(2.18)	73.95(1.61)	73.95(2.98)	73.95(2.56)	77.08(4.56)	76.56(3.71)	73.42(2.11)
Liver Disorders	**70.08(3.49)** •	56.65(3.28)	56.97(3.76)	61.62(3.81)	58.13(4.01)	58.13(3.27)	58.00(4.25)	58.00(4.25)	65.23(2.29)
Breast (WDBC)	97.41(1.07)	97.59(1.10)	97.18(1.02)	**97.88(0.78)**	97.88(1.58)	97.88(1.58)	95.77(2.38)	97.18(1.38)	95.42(0.89)
Blood Transfusion	**79.14(1.03)** •	77.65(3.62)	77.12(3.36)	73.40(1.16)	75.00(2.87)	75.00(2.36)	76.06(2.68)	73.40(4.19)	77.54(2.03)
Banana	91.78(2.68)	93.08(1.67)	92.28(2.87)	**95.21(3.18)**	95.21(2.15)	95.21(2.15)	80.31(7.20)	88.29(3.38)	90.73(3.45)
Vehicle	82.75(1.70)	83.01(1.54)	82.54(1.70)	82.54(4.05)	80.33(1.84)	81.50(3.24)	74.05(6.65)	**84.90(2.01)**	80.09(1.47)
Lithuanian Classes	93.18(1.32)	93.33(2.50)	95.33(2.64)	98.00(2.46)	85.71(2.20)	**98.66(3.85)**	88.33(3.89)	86.00(3.33)	89.33(2.29)
Sonar	**80.55(5.39)**	74.95(2.79)	76.69(1.94)	78.52(3.86)	76.51(2.06)	74.52(1.54)	76.91(3.20)	76.56(2.58)	75.72(2.82)
Ionosphere	**89.94(1.96)**	89.77(3.07)	87.50(1.67)	88.63(2.12)	88.00(1.98)	88.63(1.98)	81.81(2.52)	87.50(2.15)	85.71(5.52)
Wine	**99.25(1.11)** •	97.77(1.53)	97.77(1.62)	95.55(1.77)	85.71(2.25)	88.88(3.02)	88.88(3.02)	97.77(1.62)	95.50(4.14)
Haberman	**76.71(1.86)**	71.23(4.16)	73.68(2.27)	72.36(2.41)	70.16(3.56)	69.73(4.17)	73.68(3.61)	67.10(7.65)	75.00(3.40)
Cardiotocography (CTG)	84.62(1.08)	**86.27(1.57)**	85.71(2.20)	**86.27(1.57)**	86.65(2.35)	86.65(2.35)	86.27(1.78)	85.71(2.21)	86.02(3.04)
Vertebral Column	86.89(2.46)	85.89(2.27)	**87.17(2.24)**	82.05(3.20)	85.00(3.25)	85.89(3.74)	77.94(5.80)	84.61(3.95)	86.98(3.21)
Steel Plate Faults	67.21(1.20)	67.35(2.01)	67.96(1.98)	68.17(1.59)	66.00(1.69)	66.52(1.65)	67.76(1.54)	68.17(1.59)	**68.57(1.85)**
WDG V1	**84.56(0.36)**	84.01(1.10)	84.01(1.10)	84.01(1.10)	80.50(0.56)	80.50(0.56)	79.95(0.85)	78.75(1.35)	84.21(0.45)
Ecoli	77.25(3.52)	76.47(2.76)	75.29(3.41)	75.29(3.41)	75.29(3.41)	75.29(3.41)	76.47(3.06)	76.47(3.06)	**80.00(4.25)** •
Glass	66.87(2.99)	57.65(5.85)	61.00(2.88)	55.32(4.98)	59.45(2.65)	57.60(3.65)	57.60(3.65)	**67.92(3.24)**	62.45(3.65)
ILPD	69.40(1.64)	67.12(2.35)	69.17(1.58)	67.12(2.35)	**69.86(2.20)**	**69.86(2.20)**	**69.86(2.20)**	68.49(3.27)	68.49(3.27)
Adult	**87.15(2.43)** •	80.34(1.57)	79.76(2.26)	80.34(1.57)	83.58(2.32)	82.08(2.42)	80.34(1.32)	78.61(3.32)	79.76(2.26)
Weaning	**87.15(2.43)** •	78.94(1.25)	81.57(3.65)	82.89(3.52)	77.63(2.35)	77.63(2.35)	80.26(1.52)	81.57(2.86)	82.57(3.33)
Laryngeal1	**79.67(3.78)** •	77.35(4.45)	77.35(4.45)	77.35(4.45)	77.35(4.45)	77.35(4.45)	75.47(5.55)	77.35(4.45)	77.35(4.45)
Laryngeal3	72.65(2.17)	70.78(3.68)	72.03(1.89)	72.03(1.89)	72.90(2.30)	71.91(1.01)	61.79(7.80)	71.91(1.01)	**73.03(1.89)**
Thyroid	**96.78(0.87)**	95.95(1.25)	95.95(1.25)	95.37(2.02)	95.95(1.25)	95.95(1.25)	94.79(2.30)	95.95(1.25)	95.95(1.25)
German credit	**75.55(1.31)** •	72.80(1.95)	72.40(1.80)	74.00(3.30)	73.33(2.85)	71.20(2.52)	71.20(2.52)	73.60(3.30)	73.60(3.30)
Heart	84.80(3.36)	83.82(4.05)	83.82(4.05)	83.82(4.05)	85.29(3.69)	85.29(3.69)	**86.76(5.50)**	83.82(4.05)	83.82(4.05)
Satimage	**96.21(0.87)**	95.35(1.23)	95.86(1.07)	93.00(2.90)	95.00(1.40)	94.14(1.07)	93.28(2.10)	95.86(1.07)	95.86(1.07)
Phoneme	**80.35(2.58)**	79.06(2.50)	78.92(3.33)	79.06(2.50)	78.84(2.53)	78.84(2.53)	64.94(7.75)	73.37(5.55)	78.92(3.33)
Monk2	**83.24(2.19)** •	80.55(3.32)	77.77(4.25)	75.92(4.25)	74.07(6.60)	74.07(6.60)	75.92(5.65)	74.07(6.60)	80.55(3.32)
Mammographic	**84.82(1.55)** •	82.21(2.27)	82.21(2.27)	80.28(3.02)	82.21(2.27	82.21(2.27)	75.55(5.50)	81.25(2.07)	82.21(2.27)
MAGIC Gamma Telescope	**84.35(3.27)** •	80,03(3.25)	79,99(3.55)	81.73(3.27)	81,53(3.35)	81,16(3.00)	73,13(6.35)	75,91(5.35)	80,03(3.25)

We split the results in two tables: Table 2.3 shows a comparison with the proposed META-DES against the eight state-of-the-art dynamic selection techniques considered. A comparison of the META-DES against static combination rules is shown in Table 2.4. Each pair of results is compared using the Kruskal-Wallis non-parametric statistical test, with a 95% confidence interval. The best results are in bold. Results that are significantly better ($p < 0.05$) are marked with a •.

We can see in Table 2.3 the proposed META-DES achieves results that are either superior or equivalent to the state-of-the-art DES techniques in 25 datasets (84% of the datasets). In addition, the META-DES achieved the highest recognition performance for 18 datasets, which corresponds to 60% of the datasets considered. Only for the Ecoli, Heart, Vehicle, Banana and Lithuanian datasets (16% of the datasets) the recognition rates of the proposed META-DES

Table 2.4 Mean and standard deviation results of the accuracy obtained for the proposed DES and static ensemble combination. A pool of 100 Perceptrons as base classifier is used for all techniques The best results are in bold. Results that are significantly better ($p < 0.05$) are marked with a •

Database	META-DES	Single Best	Bagging	AdaBoost	Static Selection	Oracle
Pima	**79.03(2.24)** •	73.57(1.49)	73.28(2.08)	72.52(2.48)	72.86(4.78)	95.10(1.19)
Liver Disorders	**70.08(3.49)** •	65.38(3.47)	62.76(4.81)	64.65(3.26)	59.18(7.02)	93.07(2.41)
Breast (WDBC)	97.41(1.07)	97.04(0.74)	96.35(1.14)	**98.24(0.89)**	96.83(1.00)	99.13(0.52)
Blood Transfusion	**79.14(1.03)** •	75.07(1.83)	75.24(1.67)	75.18(2.08)	75.74(2.23)	94.20(2.08)
Banana	**91.78(2.68)**	84.07(2.22)	81.43(3.92)	81.61(2.42)	81.35(4.28)	94.75(2.09)
Vehicle	**82.75(1.70)**	81.87(1.47)	82.18(1.31)	80.56(4.51)	81.65(1.48)	96.80(0.94)
Lithuanian Classes	**93.18(1.32)** •	84.35(2.04)	82.33(4.81)	82.70(4.55)	82.66(2.45)	98.35 (0.57)
Sonar	**80.55(5.39)**	78.21(2.36)	76.66(2.36)	74.95(5.21)	79.03(6.50)	94.46(1.63)
Ionosphere	**89.94(1.96)**	87.29(2.28)	86.75(2.75)	86.75(2.34)	87.50(2.23)	96.20(1.72)
Wine	**99.25(1.11)**	96.70(1.46)	95.56(1.96)	**99.20(0.76)**	96.88(1.80)	100.00(0.01)
Haberman	**76.71(1.86)**	75.65(2.68)	72.63(3.45)	75.26(3.38)	73.15(3.68)	97.36(3.34)
Cardiotocography (CTG)	**84.62(1.08)**	84.21(1.10)	**84.54(1.46)**	83.06(1.23)	84.04(2.02)	93.08(1.46)
Vertebral Column	**86.89(2.46)**	82.04(2.17)	85.89(3.47)	83.22(3.59)	84.27(3.24)	97.40(0.54)
Steel Plate Faults	**67.21(1.20)**	66.05(1.98)	67.02(1.98)	66.57(1.06)	67.22(1.64)	88.72(1.89)
WDG V1	**84.56(0.36)**	83.17(0.76)	84.36(0.56)	84.04(0.37)	84.23(0.53)	97.82(0.54)
Ecoli	**77.25(3.52)** •	69.35(2.68)	72.22(3.65)	70.32(3.65)	67.80(4.60)	91.54(1.55)
Glass	**66.87(2.99)** •	52.92(4.53)	62.64(5.61)	55.89(3.25)	57.16(4.17)	90.65(0.00)
ILPD	**69.40(1.64)**	67.53(2.83)	67.20(2.35)	**69.38(4.28)**	67.26(1.04)	99.10(0.72)
Adult	**87.15(2.43)** •	83.64(3.34)	85.60(2.27)	83.58(2.91)	84.37(2.79)	95.59(0.39)
Weaning	**79.67(3.78)** •	74.86(4.78)	76.31(4.06)	74.47(3.68)	76.89(3.15)	92.10(0.92)
Laryngeal1	**83.43(4.50)**	80.18(5.51)	81.32(3.82)	79.81(3.88)	80.75(4.93)	98.86(0.98)
Laryngeal3	**72.65(2.17)**	68.42(3.24)	67.13(2.47)	62.32(2.57)	71.23(3.18)	100(0.00)
Thyroid	**96.78(0.87)**	95.15(1.74)	95.25(1.11)	96.01(0.74)	96.24(1.25)	99.88(0.36)
German credit	**75.55(2.31)**	71.16(2.39)	74.76(2.73)	72.96(1.25)	73.60(2.69)	99.12(0.70)
Heart	**84.80(3.36)**	80.26(3.58)	82.50(4.60)	81.61(5.01)	82.05(3.72)	95.90(1.02)
Satimage	**96.21(0.87)**	94.52(0.96)	95.23(0.87)	95.43(0.92)	95.31(0.92)	98.69(0.87)
Phoneme	**80.35(2.58)** •	75.87(1.33)	72.60(2.33)	75.90(1.06)	72.70(2.32)	99.34(0.24)
Monk2	**83.24(2.19)**	79.25(3.78)	79.18(2.57)	80.27(2.76)	80.55(3.59)	98.98(1.19)
Mammographic	84.82(1.55)	83.60(1.85)	**85.27(1.85)**	83.07(3.03)	84.23(2.14)	99.59(0.15)
MAGIC Gamma Telescope	84.35(3.27)	80.27(3.50)	81.24(2.22)	**87.35(1.45)** •	85.25(3.25)	95.35(0.68)

framework presented is statistically inferior to the best result achieved by state-of-the-art DES techniques.

For the 12 datasets where the proposed META-DES did not achieved the highest recognition rate (WDBC, Banana, Vehicle, Lithuanian, Cardiotocography, Vertebral column, Steel plate faults, Ecoli, Glass, ILPD, Laryngeal3 and Heart) we can see that each DES technique presented the best accuracy for different datasets (as shown in Figure 2.5). The KNOP achieves the best results for three datasets (Ecoli, Steel plate faults and Laryngeal3), the MCB for two datasets (Vehicle and Glass), the DES-FA for 3 datasets (Banana, Breast cancer and Cardiotocography) and so forth. This can be explained by the "no free lunch" theorem. There is no criterion to estimate the competence of base classifiers that dominates all other when compared with several classification problems. Since the proposed META-DES uses a combination of five different criteria as meta-features, even though one criterion might fail, the

system can still achieve a good performance as other meta-features are also considered by the selection scheme. In this way, a more robust DES technique is achieved.

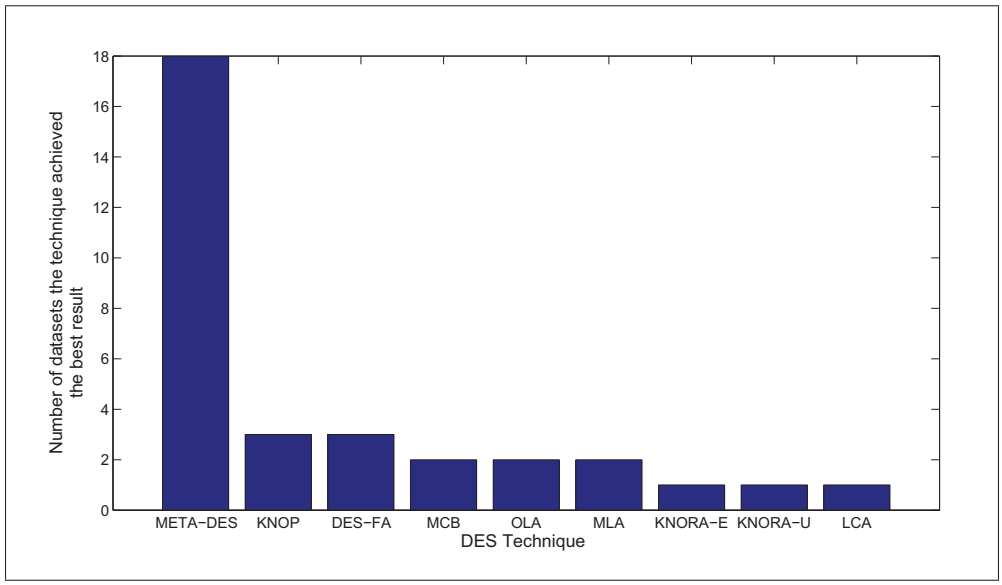


Figure 2.5 Bar plot showing the number of datasets that each DES technique presented the highest recognition accuracy

Moreover, another advantage of the proposed META-DES framework comes from the fact that several meta-feature vectors are generated for each training sample in the meta-training phase (Section 3.2.2). For instance, consider that 200 training samples are available for the meta-training stage ($N = 200$), if the pool C is composed of 100 weak classifiers ($M = 100$), the meta-training dataset is the number of training samples $N \times$ the number classifiers in the pool M, $N \times M = 20.000$. Hence, there is more data to train the meta-classifier λ than for the generation of the pool of classifiers C itself. Even though the classification problem may be ill-defined, due to the size of the training set, using the proposed framework we can overcome this limitation since the size of the meta-problem is up to 100 times bigger than the classification problem. So, our proposed framework has more data to estimate the level of competence of base classifiers than the other DES methods, where only the training or validation data is available. This fact can be observed by the results obtained for datasets with less than 500 samples for training,

such as, Liver Disorders, Sonar, Weaning and Ionosphere where recognition accuracy of the META-DES is statistically superior for those small size problems.

When compared against static ensemble techniques Table 2.4, the proposed META-DES achieves the highest recognition accuracy for 24 out of 30 datasets. This can be explained by the fact that the majority of datasets considered are ill-defined. Hence, the results found in this paper also support the claim made by Cavalin et al. [16] that DES techniques outperform static methods for ill-defined problems.

We can thus answer the research question posed in this paper: Can the use of meta-features lead to a more robust dynamic selection technique? As the proposed system achieved better recognition rates in the majority of datasets the use of multiple properties from the classification environment as meta-features indeed leads to a more robust dynamic ensemble selection technique.

2.5 Conclusion

In this chapter, we presented a novel DES technique in a meta-learning framework. The framework is based on two environments: the classification environment, in which the input features are mapped into a set of class labels, and the meta-classification environment, in which different properties from the classification environment, such as the classifier accuracy in the feature space or the consensus in the decision space, are extracted from the training data and encoded as meta-features. Five sets of meta-features are proposed. Each set corresponding to a different dynamic selection criterion. These meta-features are used to train a meta-classifier which can estimate whether a base classifier is competent enough to classify a given input sample. With the arrival of new test data, the meta-features are extracted using the test data as reference, and used as input to the meta-classifier. The meta-classifier decides whether the base classifier is competent enough to classify the test sample.

Experiments were conducted using 30 classification datasets coming from five different data repositories (UCI, KEEL, STATLOG, LKC and ELENA) and compared against eight state-of-

the-art dynamic selection techniques (each technique based on a single criterion to measure the level of competence of a base classifier), as well as five classical static combination methods. Experimental results show the proposed META-DES achieved the highest classification accuracy in the majority of datasets, which can be explained by the fact that the proposed META-DES framework is based on five different DES criteria. Even though one criterion might fail, the system can still achieve a good performance as other criteria are also considered in order to perform the ensemble selection. In this way, a more robust DES technique is achieved.

In addition, we observed a significant improvement in performance for datasets with critical training size samples. This gain in accuracy can be explained by the fact that during the Meta-Training phase of the framework, each training sample generates several meta-feature vectors for the training of the meta-classifier. Hence, the proposed framework has more data to train the meta-classifier and consequently to estimate the level of competence of base classifiers than the current state-of-the-art DES methods, where only the training or validation data is available.

Future works on this topic will involve:

a. The definition of new sets of meta-features to better estimate the level of competence of the base classifiers.

b. The selection of meta-features based on optimization algorithms in order to improve the performance of the meta-classifier, and consequently, the accuracy of the DES system.

c. The evaluation of different training scenarios for the meta-classifier.

In the next chapter a detailed analysis of the META-DES framework is conducted in order to fully understand the impact of each set of meta-features for the definition of a competent classifiers as well as other aspects of the framework in the training and test phases.

CHAPTER 3

A DEEP ANALYSIS OF THE META-DES FRAMEWORK FOR DYNAMIC SELECTION OF ENSEMBLE OF CLASSIFIERS

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Published in « arXiv » 2015.

Abstract

Dynamic ensemble selection (DES) techniques work by estimating the level of competence of each classifier from a pool of classifiers. Only the most competent ones are selected to classify a given test sample. Hence, the key issue in DES is the criterion used to estimate the level of competence of the classifiers in predicting the label of a given test sample. In order to perform a more robust ensemble selection, we proposed the META-DES framework using meta-learning, where multiple criteria are encoded as meta-features and are passed down to a meta-classifier that is trained to estimate the competence level of a given classifier. In this technical report, we present a step-by-step analysis of each phase of the framework during training and test. We show how each set of meta-features is extracted as well as their impact on the estimation of the competence level of the base classifier. Moreover, an analysis of the impact of several factors in the system performance, such as the number of classifiers in the pool, the use of different linear base classifiers, as well as the size of the validation data. We show that using the dynamic selection of linear classifiers through the META-DES framework, we can solve complex non-linear classification problems where other combination techniques such as AdaBoost cannot.

3.1 Introduction

Multiple Classifier Systems (MCS) aim to combine classifiers in order to increase the recognition accuracy in pattern recognition systems [24; 9]. MCS are composed of three phases [1]: (1) Generation, (2) Selection, and (3) Integration. In the first phase, a pool of classifiers is generated. In the second phase, a single classifier or a subset having the best classifiers of the pool is(are) selected. We refer to the subset of classifiers as the Ensemble of Classifiers (EoC). In the last phase, integration, the predictions of the selected classifiers are combined to obtain the final decision [24].

Recent works in MCS have shown that dynamic ensemble selection (DES) techniques achieve higher classification accuracy when compared to static ones [1; 2; 14]. This is specially true for ill-defined problems, i.e., for problems where the size of the training data is small, and there are not enough data available to train the classifiers [16; 17]. The key issue in DES is to define a criterion to measure the level of competence of a base classifier. Most DES techniques [14; 22; 21; 20] estimate the classifiers' local accuracy in small regions of the feature space surrounding the query instance, called the region of competence, as a search criterion for estimating the competence level of the base classifier. However, in our previous work [20], we demonstrated that the use of local accuracy estimates alone is insufficient to provide higher classification performance. In addition, a dissimilarity analysis among eight dynamic selection techniques, performed in [55], indicates that techniques based on different criteria for estimating the competence level of base classifiers yields different results.

To tackle this issue, in [2] we proposed a novel DES framework, called META-DES, in which multiple criteria regarding the behavior of a base classifier are used to compute its level of competence. The framework is based on two environments: the classification environment, in which the input features are mapped into a set of class labels, and the meta-classification environment, where several properties from the classification environment, such as the classifier accuracy in a local region of the feature space, are extracted from the training data and encoded as meta-features. Given a test data, the meta-features are extracted using the test data

as reference, and used as input to the meta-classifier. The meta-classifier decides whether the base classifier is competent enough to classify the test sample.

One interesting properties of the META-DES framework is that it obtains higher classification accuracy using only linear classifiers. In this work, we perform a deep analysis of the training and classification steps of the META-DES framework. We perform step-by-step examples in order to show the influence of different sets of meta-features used to better estimate the competence of the base classifier. The analysis is conducted using the P2 problem [67; 68] which is a two-class non-linear problem with a complex decision boundary. Furthermore, the two-classes of the P2 problem have multiple class means, making it a difficult classification problem.

The following points of the META-DES framework are studied:

- The use of weak, linear classifiers in the pool. In this work we consider both Perceptrons and Decision Stumps as base classifiers.
- The influence of each set of meta-features for estimating the competence of a base classifier.
- The influence of the dynamic selection set (DSEL)[1] in the recognition rate. The dynamic selection data is used in order to extract the meta-features.
- The influence of the size of the Pool in the classification accuracy of the META-DES framework.

The contributions of this work are as follows:

- It shows that using dynamic selection of linear and weak classifiers, such as Perceptrons and Decision stumps, we can solve problems with complex decision boundaries, including classification problems with multiple class centers.

[1] DSEL is often called validation data in several dynamic selection techniques.

- It allows an understanding of why the META-DES framework achieves high recognition accuracy using only linear classifiers. In previous works, the META-DES was presented as a black box system. In this work, we use a step-by-step example to illustrate how the framework is able to select the competent classifiers based on the five defined sets of meta-features.

- It compares the dynamic selection of linear classifiers against static combination rules such as AdaBoost, as well as classical single classifier models, such as Multi-Layer Perceptron neural networks, Random Forest, and Support Vector Machnies (SVMs).

This document is organized as follows. Theoretical aspects of dynamic selection are introduced in Section 3.2. The META-DES framework is presented in Section 3.3. An illustrative example of the META-DES is presented in Section 3.4. Experiments are carried out in Section 3.5. Conclusions are given in the last section.

3.2 Why does dynamic selection of linear classifiers work?

Let $C = \{c_1, \ldots, c_M\}$ (M is the size of the pool of classifiers) be the pool of classifiers and c_i a base classifier belonging to the pool C. The goal of dynamic selection is to find an ensemble of classifiers $C' \subset C$ that has the best classifiers to classify a given test sample $\mathbf{x}_j$. DES techniques rely on the assumption that each base classifier is an expert in a different local region of the feature space [27]. Only the classifiers that attain a certain competence level, according to a selection criterion, are selected to predict the label of $\mathbf{x}_j$. This is a different strategy when compared with static selection, where the ensemble of classifiers C' is selected during the training phase, and considering the global performance of the base classifiers over a validation dataset [10; 11; 12; 13].

When dealing with dynamic selection, we aim to select the appropriate classifier(s) for a specific test sample $\mathbf{x}_j$, rather than find the best decision border separating the classes. This is a different concept, as compared to classical classification models, such as Support Vector Machines (SVM) or Multi-Layer Perceptrons (MLP) Neural Networks in the sense that these

classifiers search for the best separation between the classes during the training stages. This is an important property of dynamic selection techniques, which makes them suitable for solving problems that are ill-defined, i.e., when there is not enough data available to train a strong classifier having a lot of parameters to learn [16]. In addition, due to insufficient training data, the distribution of the training data may not adequately represent the real distribution of the problem. Consequently, the classifiers cannot learn the separation between the classes.

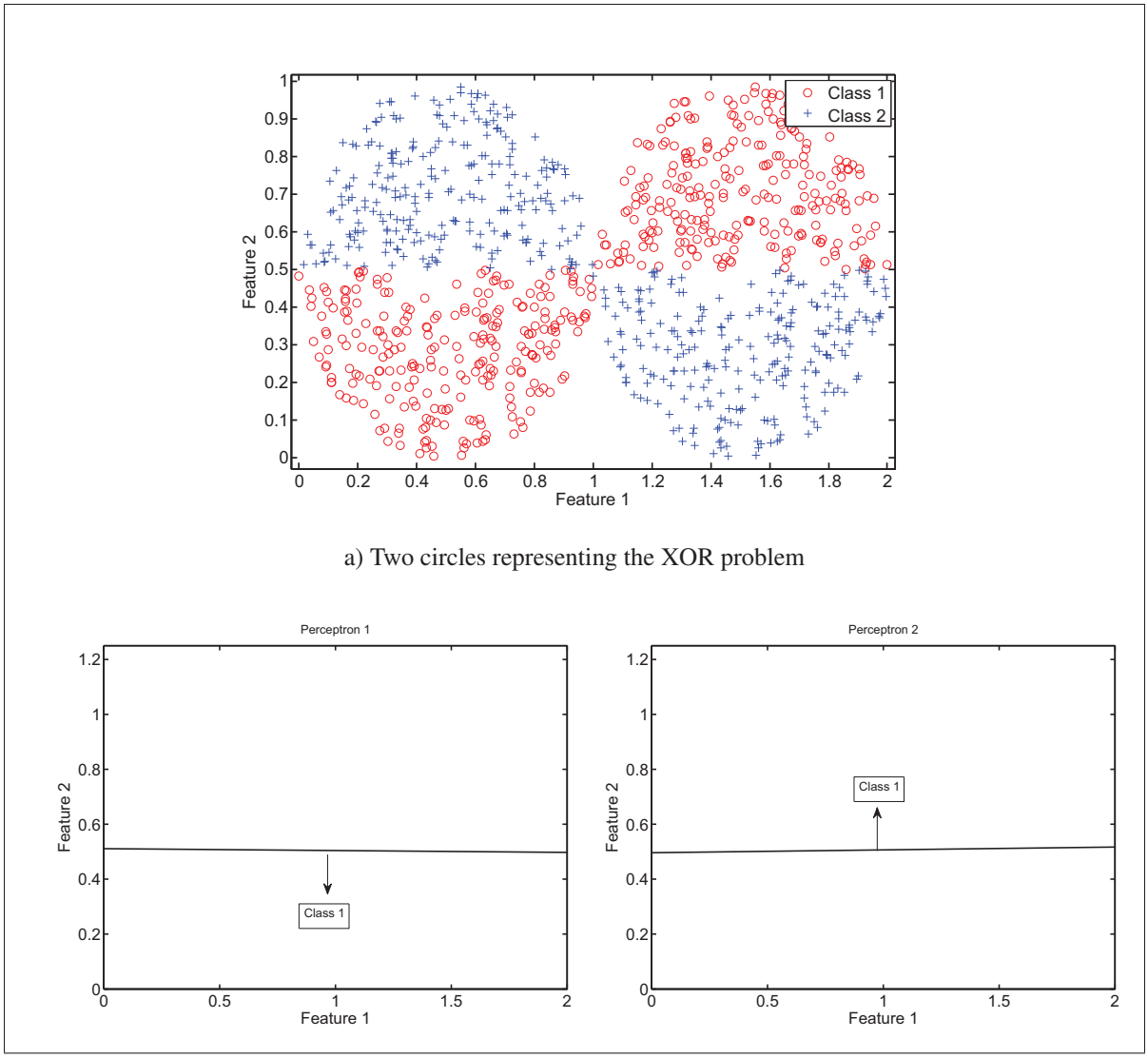
Let us consider, for instance, two circles representing the exclusive or XOR problem. The problem is generated with 1000 data points, 500 for each class (Figure 3.1 (a)). Two linear classifiers trained for this problem (two Perceptrons) c_1 and c_2, both with an individual accuracy of 50%. The decisions of c_1 and c_2 are shown in (Figure 3.1 (b) and (c) respectively).

Static combination rules, such as majority voting or averaging are useless in this case since the base classifiers always yield opposite decisions, i.e., for any query sample $\mathbf{x}_j$, if c_1 predicts that $\mathbf{x}_j$ belongs to class 1, c_2 will predict that $\mathbf{x}_j$ belongs to class 2 and vice versa. There is never a consensus between the decisions obtained by these two classifiers.

Considering the same data, it is possible to split the feature space into four local regions (Figure 3.2): Q1, Q2, Q3 and Q4.

Using dynamic selection, it is possible to obtain a 100% accuracy rate using only these two classifiers. Given a query instance $\mathbf{x}_j$, the system first checks the competence of each classifier in the pool. Only the classifier(s) with the highest competence are selected. Classifiers that are not experts in the local region will not influence the ensemble decision.

Given a query sample $\mathbf{x}_j$ to be classified, using dynamic selection, the classification is performed as follows(Equation 3.1):



a) Two circles representing the XOR problem

Figure 3.1 (a) The two circles data generated with 1000 data points, 500 samples for each class; (b) illustrates the decision made by the Perceptron c_1; (c) shows the decision made by the Perceptron c_2

$$\begin{cases} \text{If } \mathbf{x}_j \in \text{Q1} & \text{Select } c_1 \\ \text{If } \mathbf{x}_j \in \text{Q2} & \text{Select } c_2 \\ \text{If } \mathbf{x}_j \in \text{Q3} & \text{Select } c_2 \\ \text{If } \mathbf{x}_j \in \text{Q4} & \text{Select } c_1 \end{cases} \tag{3.1}$$

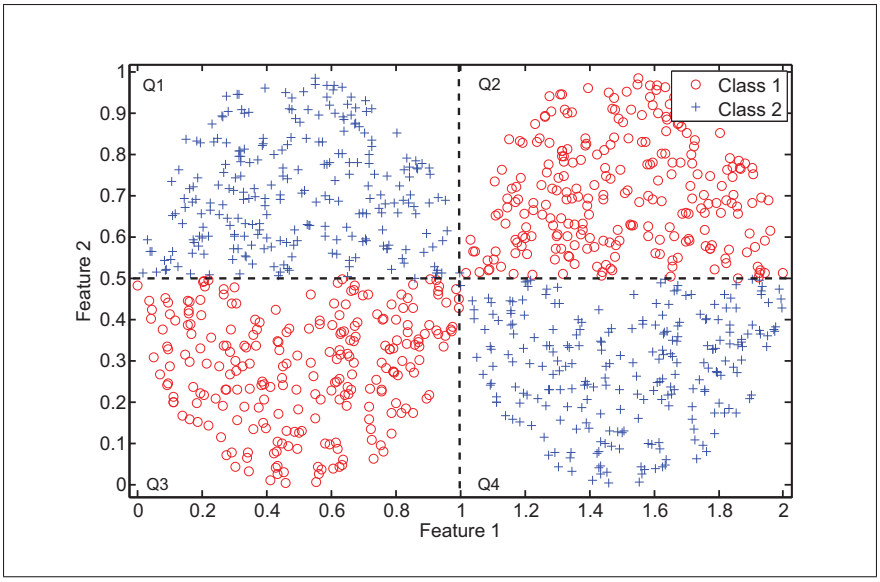


Figure 3.2 The two circles data divided into four regions

The key issue in DES is to define a criterion to measure the level of competence of a base classifier. Most DES techniques [14; 22; 21; 20; 28; 29; 30; 31] use estimates of the classifiers' local accuracy in small regions of the feature space surrounding the query instance as a search criterion to perform the ensemble selection. There are other criteria, such as the degree of consensus, in the ensemble [15], probabilistic models applied to the classifier outputs [18] and decision templates [16; 17]. A recent survey on dynamic selection [1] covers all the DES criteria used by different techniques.

In [2; 36], we proposed a novel DES framework in which multiple criteria regarding the behavior of a base classifier are used to have a better estimation of its level of competence. The META-DES framework is presented in the following sections.

3.3 The META-DES Framework

The META-DES framework is based on the assumption that the dynamic ensemble selection problem can be considered as a meta-problem. This meta-problem uses different criteria regarding the behavior of a base classifier c_i, in order to decide whether it is competent enough to classify a given test sample $\mathbf{x}_j$. The meta-problem is defined as follows [2]:

- The **meta-classes** of this meta-problem are either "competent" (1) or "incompetent" (0) to classify $\mathbf{x}_j$.
- Each set of **meta-features** f_i corresponds to a different criterion for measuring the level of competence of a base classifier.
- The meta-features are encoded into a **meta-features vector** $v_{i,j}$.
- A **meta-classifier** λ is trained based on the meta-features $v_{i,j}$ to predict whether or not c_i will achieve the correct prediction for $\mathbf{x}_j$, i.e., if it is competent enough to classify $\mathbf{x}_j$

A general overview of the META-DES framework is depicted in Figure 3.3. It is divided into three phases: Overproduction, Meta-training and Generalization.

3.3.1 Overproduction

In this step, the pool of classifiers $C = \{c_1, \ldots, c_M\}$, where M is the pool size, is generated using the training dataset $\mathcal{T}$. The Bagging technique [3] is used in this work in order to build a diverse pool of classifiers.

3.3.2 Meta-Training

In this phase, the meta-features are computed and used to train the meta-classifier λ. As shown in Figure 3.3, the meta-training stage consists of three steps: sample selection, meta-features extraction process and meta-training. A different dataset $\mathcal{T}_\lambda$ is used in this phase to prevent overfitting.

3.3.2.1 Sample selection

We decided to focus the training of λ on cases in which the extent of consensus of the pool is low. This decision was based on the observations made in [15; 16] the main issues in dynamic ensemble selection occur when classifying testing instances where the degree of consensus

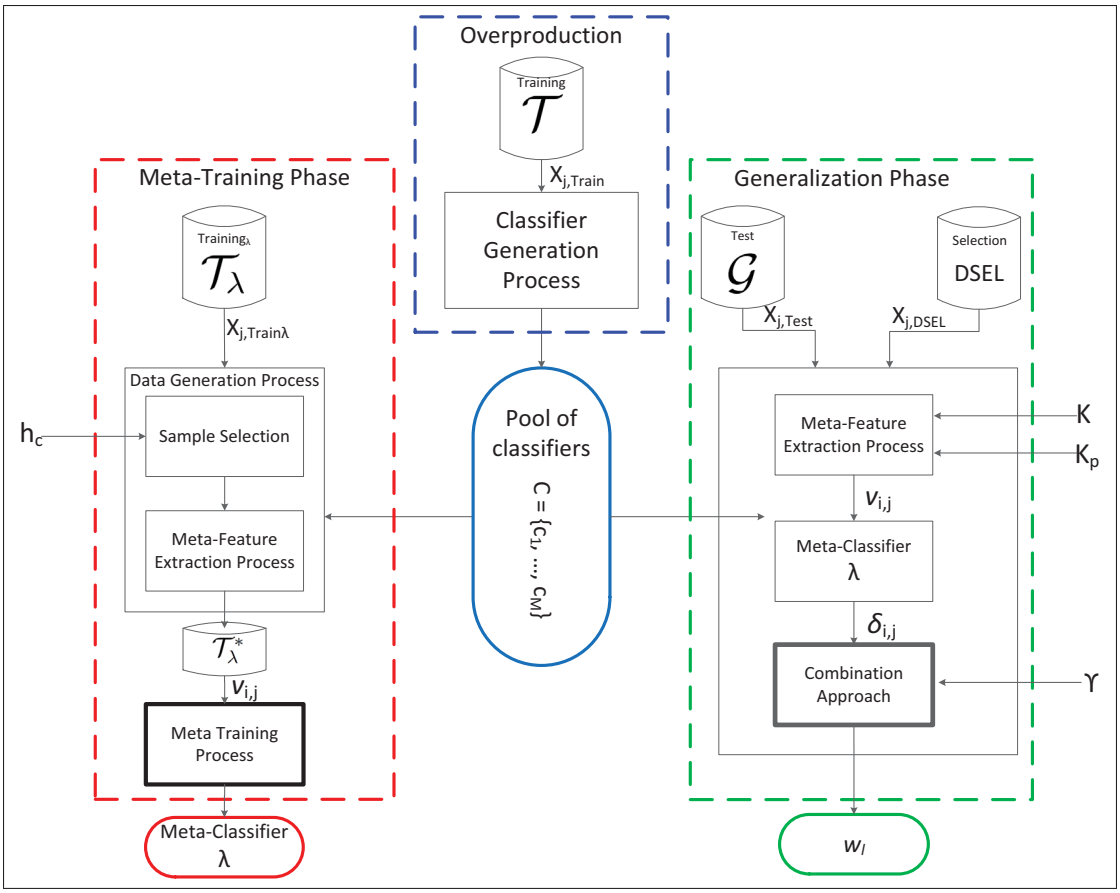


Figure 3.3 Overview of the proposed framework. It is divided into three steps 1) Overproduction, where the pool of classifiers $C = \{c_1, \ldots, c_M\}$ is generated, 2) The training of the selector λ (meta-classifier), and 3) The generalization phase where the level of competence $\delta_{i,j}$ of each base classifier c_i is calculated specifically for each new test sample $\mathbf{x}_{j,test}$. Then, the level of competence $\delta_{i,j}$ is used by the combination approach to predict the label w_l of the test sample $\mathbf{x}_{j,test}$. Three combination approaches are considered: Dynamic selection (META-DES.S), Dynamic weighting (META-DES.W) and Hybrid (META-DES.H). h_C, K, K_p and Υ are the hyper-parameters required by the proposed system [Adapted from [2]]

among the pool of classifiers is low, i.e., when the number of votes from the winning class is close to or even equal to the number of votes from the second class. We employ a sample selection mechanism based on a threshold h_C, called the consensus threshold. For each $\mathbf{x}_{j,train_\lambda} \in \mathcal{T}_\lambda$, the degree of consensus of the pool, denoted by $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$, is computed. If $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$ falls below the threshold h_C, $\mathbf{x}_{j,train_\lambda}$ is passed down to the meta-features extraction process.

3.3.2.2 Meta-feature extraction

The first step in extracting the meta-features involves computing the region of competence of $\mathbf{x}_{j,train_\lambda}$, denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. The region of competence is defined in the $\mathcal{T}_\lambda$ set using the K-Nearest Neighbor algorithm. Then, $\mathbf{x}_{j,train_\lambda}$ is transformed into an output profile, $\tilde{\mathbf{x}}_{j,train_\lambda} = \left\{\tilde{\mathbf{x}}_{j,train_\lambda,1}, \tilde{\mathbf{x}}_{j,train_\lambda,2}, \ldots, \tilde{\mathbf{x}}_{j,train_\lambda,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,train_\lambda,i}$ is the decision yielded by the base classifier c_i for the sample $\mathbf{x}_{j,train_\lambda}$ [16].

The similarity between $\tilde{\mathbf{x}}_{j,train_\lambda}$ and the output profiles of the instances in $\mathcal{T}_\lambda$ is obtained through the Euclidean distance. The most similar output profiles are selected to form the set $\phi_j = \left\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{K_p}\right\}$, where each output profile $\tilde{\mathbf{x}}_k$ is associated with a label $w_{l,k}$. Next, for each base classifier $c_i \in C$, five sets of meta-features are calculated:

f_1 - **Neighbors' hard classification:** First, a vector with K elements is created. For each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of the vector is set to 1, otherwise it is 0. Thus, K meta-features are computed.

f_2 - **Posterior probability:** First, a vector with K elements is created. Then, for each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, the posterior probability of c_i, $P(w_l \mid \mathbf{x}_k)$ is computed and inserted into the k-th position of the vector. Consequently, K meta-features are computed.

f_3 - **Overall Local accuracy:** The accuracy of c_i over the whole region of competence θ_j is computed and encoded as f_3.

f_4 - **Output profiles classification:** First, a vector with K_p elements is generated. Then, for each member $\tilde{\mathbf{x}}_k$ belonging to the set of output profiles ϕ_j, if the label produced by c_i for $\mathbf{x}_k$ is equal to the label $w_{l,k}$ of $\tilde{\mathbf{x}}_k$, the k-th position of the vector is set to 1, otherwise it is 0. A total of K_p meta-features are extracted using output profiles.

f_5 - **Classifier's Confidence:** The perpendicular distance between the input sample $\mathbf{x}_{j,train_\lambda}$ and the decision boundary of the base classifier c_i is calculated and encoded as f_5. f_5 is normalized to a $[0-1]$ range using the Min-max normalization.

A vector $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ (Figure 3.4) is obtained at the end of the process. If c_i correctly classifies $\mathbf{x}_j$, the class attribute of $v_{i,j}$, $\alpha_{i,j} = 1$ (i.e., $v_{i,j}$ belongs to the meta-class "competent"), otherwise $\alpha_{i,j} = 0$. $v_{i,j}$ is stored in the meta-features dataset $\mathcal{T}_\lambda^*$ that is used to train the meta-classifier λ. Figure 3.4 illustrates the format of the meta-features vector $v_{i,j}$.

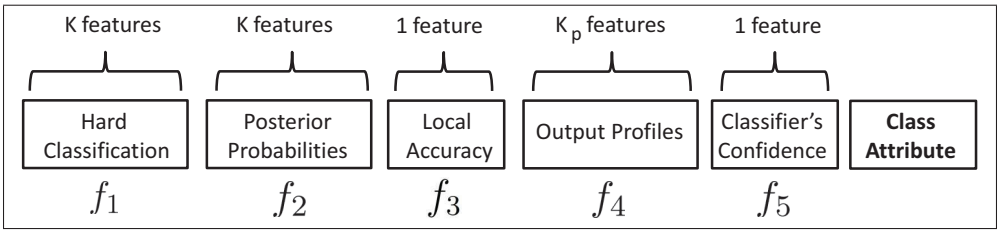


Figure 3.4 Feature Vector containing the meta-information about the behavior of a base classifier. A total of 5 different meta-features are considered. The size of the feature vector is $(2 \times K) + K_p + 2$. The class attribute indicates whether or not c_i correctly classified the input sample

3.3.2.3 Training

The last step of the meta-training phase is the training of the meta-classifier λ. In this work, we considered a Naive Bayes for the meta-classifier λ, since this classifier model presented the best classification results for the META-DES framework when compared against different classifier models, such as a Multi-Layer Perceptron Neural Network and a Random Forest classifier [69].

3.3.3 Generalization

Given the query sample $\mathbf{x}_{j,test}$, the region of competence θ_j is computed using the samples from the dynamic selection dataset *DSEL*. Following that, the output profiles $\tilde{\mathbf{x}}_{j,test}$ of the test sample, $\mathbf{x}_{j,test}$, are calculated. The set with K_p similar output profiles ϕ_j, of the query sample $\mathbf{x}_{j,test}$, is obtained through the Euclidean distance applied over the output profiles of the dynamic selection dataset, $\tilde{DSEL}$.

For each base classifier, c_i, belonging to the pool of classifiers, C, the five sets of meta-features are computed, returning the meta-features vector $v_{i,j}$. Then, $v_{i,j}$ is used as input to the meta-classifier λ. The support obtained by λ for the "competent" meta-class is computed as the level of competence, $\delta_{i,j}$, of the base classifier c_i for the classification of the test sample $\mathbf{x}_{j,test}$. As in [69], we consider a hybrid combination approach called META-DES.H. First, the base classifiers that achieve a level of competence $\delta_{i,j} > \Upsilon = 0.5$ are selected to compose the ensemble C'. Next, the decision of each selected base classifier is weighted by its level of competence.A weighted majority voting approach is used to predict the label w_l of the sample $\mathbf{x}_{j,test}$. Thus, the decisions obtained by the base classifiers that attained a higher level of competence $\delta_{i,j}$ have a greater influence in the final decision.

3.4 Why does the META-DES work: A Step-by-step example

In this section, we present a step-by-step example of the training and test phases of the META-DES framework in order to understand the mechanisms behind the META-DES, and why it achieves good generalization performance using only linear classifiers. For this example, we use the P2 problem.

3.4.1 The P2 Problem

The P2 is a two-class problem, presented by Valentini [67], in which each class is defined in multiple decision regions delimited by polynomial and trigonometric functions (Equation 3.2). As in [68], $E4$ was modified such that the area of each class was approximately equal. The P2 problem is illustrated in Figure 3.5. One can clearly see that it is impossible to solve this problem using linear classifiers. The performance of the best possible linear classifier is around 50%.

$$E1(x) = sin(x) + 5 \tag{3.2}$$

$$E2(x) = (x-2)^2 + 1 \tag{3.3}$$

$$E3(x) = -0.1 \cdot x^2 + 0.6sin(4x) + 8 \tag{3.4}$$

$$E4(x) = \frac{(x-10)^2}{2} + 7.902 \tag{3.5}$$

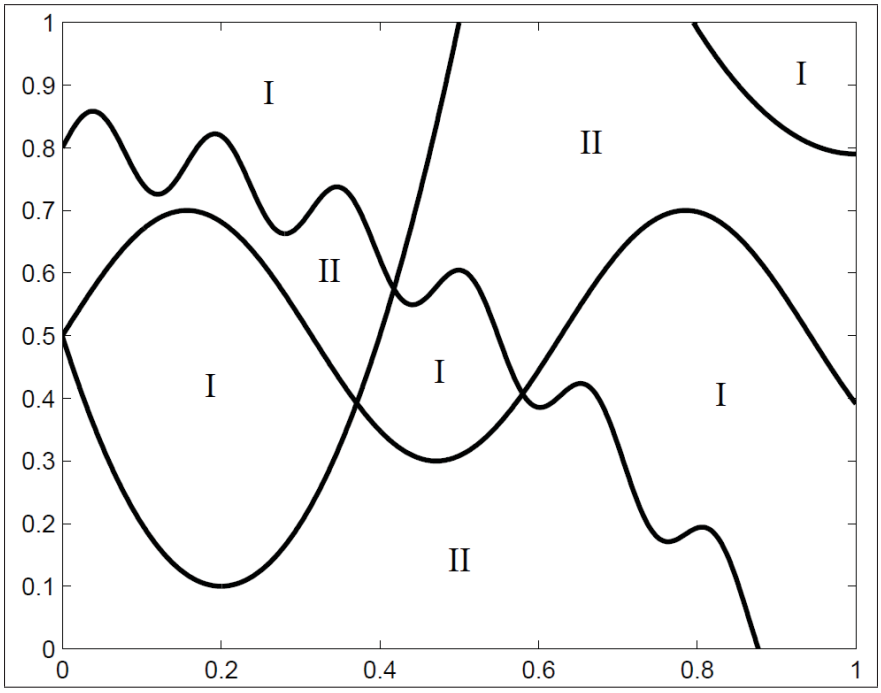


Figure 3.5 The P2 Problem. The symbols I and II represents the area of the classes 1 and 2 respectively

For this illustrative example, the P2 problem is generated as follows: 500 samples for training ($\mathcal{T}$), 500 instances for the meta-training dataset ($\mathcal{T}_\lambda$), 500 instances for the dynamic selection dataset *DSEL*, and 2000 samples for the test dataset, $\mathcal{G}$. For the sake of simplicity, we use a pool composed of 5 Perceptrons. We demonstrate that using only 5 Perceptrons it is possible to approximate the complex decision boundary of the P2 problem using the META-DES framework.

3.4.2 Overproduction

Figure 3.6 shows five Perceptrons generated using the bagging technique for the P2 problem. The arrow in each Perceptron points to the region where the classifier output is class 1 (red circle). Figure 3.7 presents the decision of each Perceptron individually.

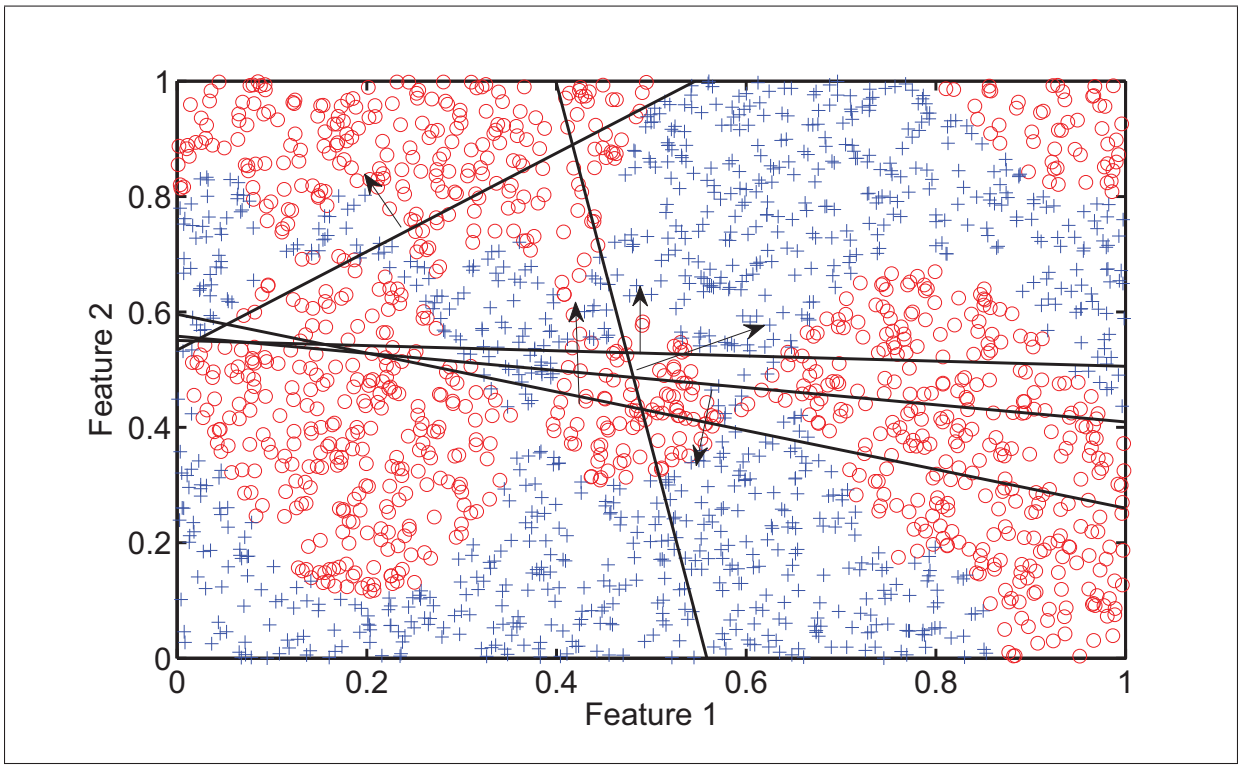
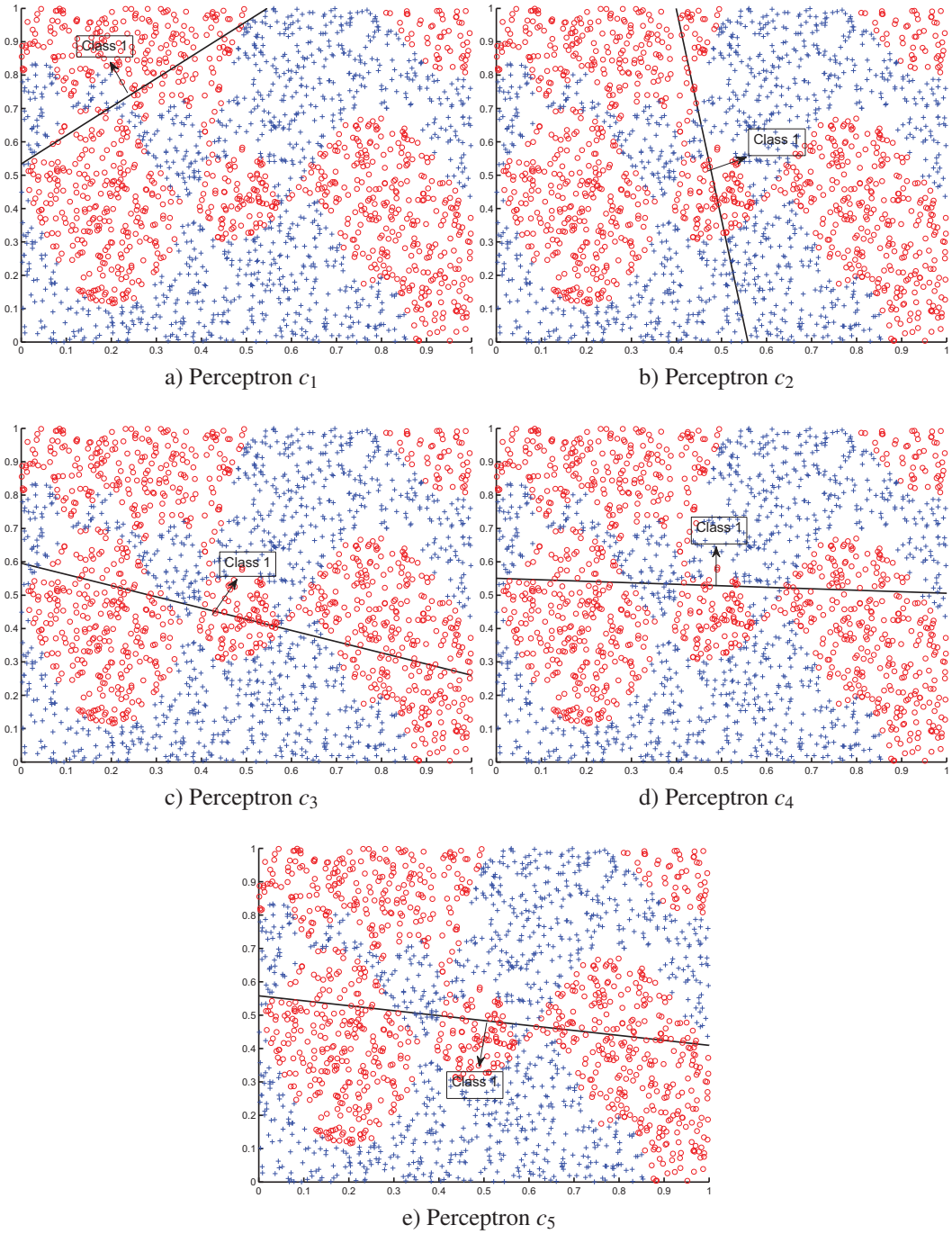


Figure 3.6 Five Perceptrons trained for the P2 Problem. The bagging technique was used to generate the pool. The arrows in each Perceptron points to the region where the classifier output is class 1 (red circle)

The best classifier of the pool (Single Best) achieves an accuracy rate of 53.5% (c_1). The performance of all other base classifiers is around the 50% mark. The Oracle result of this pool obtained a recognition rate of 99.5%. The Oracle is an abstract model defined in [32], which always selects the classifier that predicted the correct label, for the given query sample, if such a classifier exists. In other words, it represents the ideal classifier selection scheme. There is at least one base classifier that predicts the correct label for 99.5% of the test instances. The key



a) Perceptron c_1 b) Perceptron c_2

c) Perceptron c_3 d) Perceptron c_4

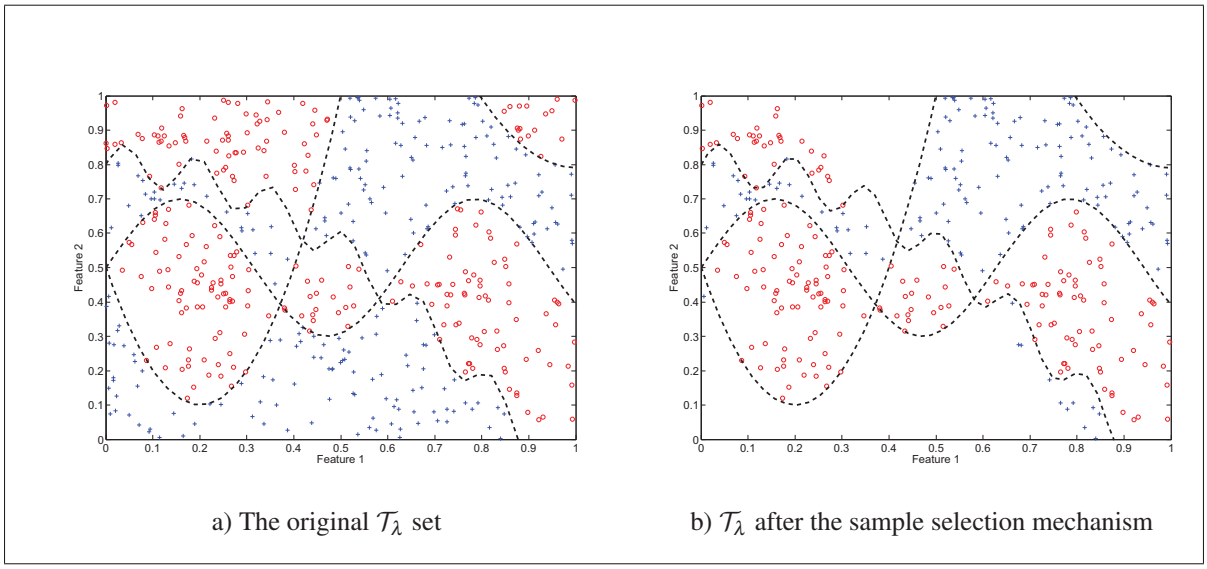
e) Perceptron c_5

Figure 3.7 Decision of each of the five Perceptrons shown separately. The arrow in each Perceptron points to the region where the classifier output is class 1 (red circle)

issue is finding the right criteria to estimate the competence of the base classifiers in order to select only the competent ones.

3.4.3 Meta-training: Sample Selection

After generating the pool of classifiers C, the next step is the sample selection mechanism for training the meta-classifier. Figure 3.8 illustrates the effect of the sample selection mechanism. As in [2; 69] the consensus threshold h_c is set at 70%. (Figure 3.8 (a)) shows the original $\mathcal{T}_\lambda$ set before the sample selection. Figure 3.8 (b) shows the samples that were selected for training the meta-classifier.



a) The original $\mathcal{T}_\lambda$ set

b) $\mathcal{T}_\lambda$ after the sample selection mechanism

Figure 3.8 (a) The original $\mathcal{T}_\lambda$ dataset generated with 500 samples (250 for each class). (b) $\mathcal{T}_\lambda$ after the sample selection mechanism was applied. 349 samples were selected

The sample selection mechanism focuses on samples whose correct labels are harder to predict, i.e., when there is no consensus between the classifiers in the pool. Samples close to the decision boundary are the ones more likely to be selected for the training of the meta-classifier. This principle is similar to the support vectors in the SVM technique, in which samples close to the decision boundary are used as support vectors to achieve a better separation between

classes. In the META-DES framework, the samples close to the decision boundary are used to train the meta-classifier, while samples that are closer to the class mean are not used for training since the majority of base classifiers can correctly classify those samples. Only the samples shown in Figure 3.8 (b) are passed down to the meta-features extraction process and are used for the training of the meta-classifier λ.

3.4.4 Classification

To illustrate the classification steps of the system we consider five testing samples in different parts of the feature space. The coordinates of the each query instance are: $\mathbf{x}_1 = [0.2, 0.9]$, $\mathbf{x}_2 = [0.2, 0.1]$, $\mathbf{x}_3 = [0.5, 0.5]$, $\mathbf{x}_4 = [0.8, 0.7]$ and $\mathbf{x}_5 = [0.9, 0.85]$. Figure 3.9 illustrates the positions of the five testing samples. $\mathbf{x}_1$ $\mathbf{x}_3$ and $\mathbf{x}_5$ belongs to class 1, $\mathbf{x}_2$ and $\mathbf{x}_4$ belongs to class 2.

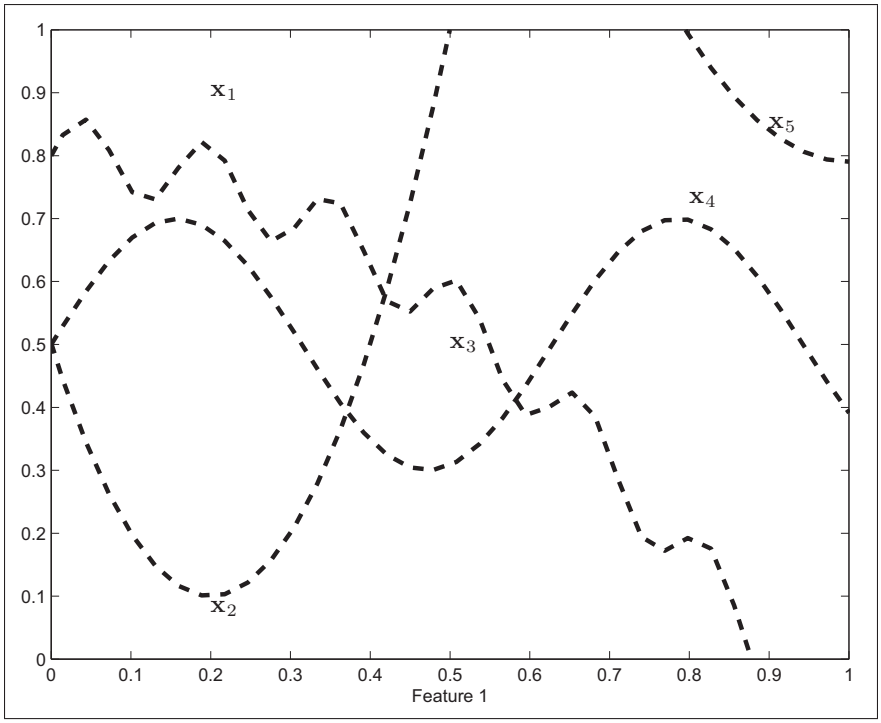


Figure 3.9 Five samples to be classified. $\mathbf{x}_1$ $\mathbf{x}_3$ and $\mathbf{x}_5$ belonging to class 1, $\mathbf{x}_2$ and $\mathbf{x}_4$ belonging to class 2

In order to compute the region of competence and extract the meta-features for the given query sample, the dynamic selection dataset (DSEL) is used in the generalization phase. The dynamic selection dataset is shown in Figure 3.10.

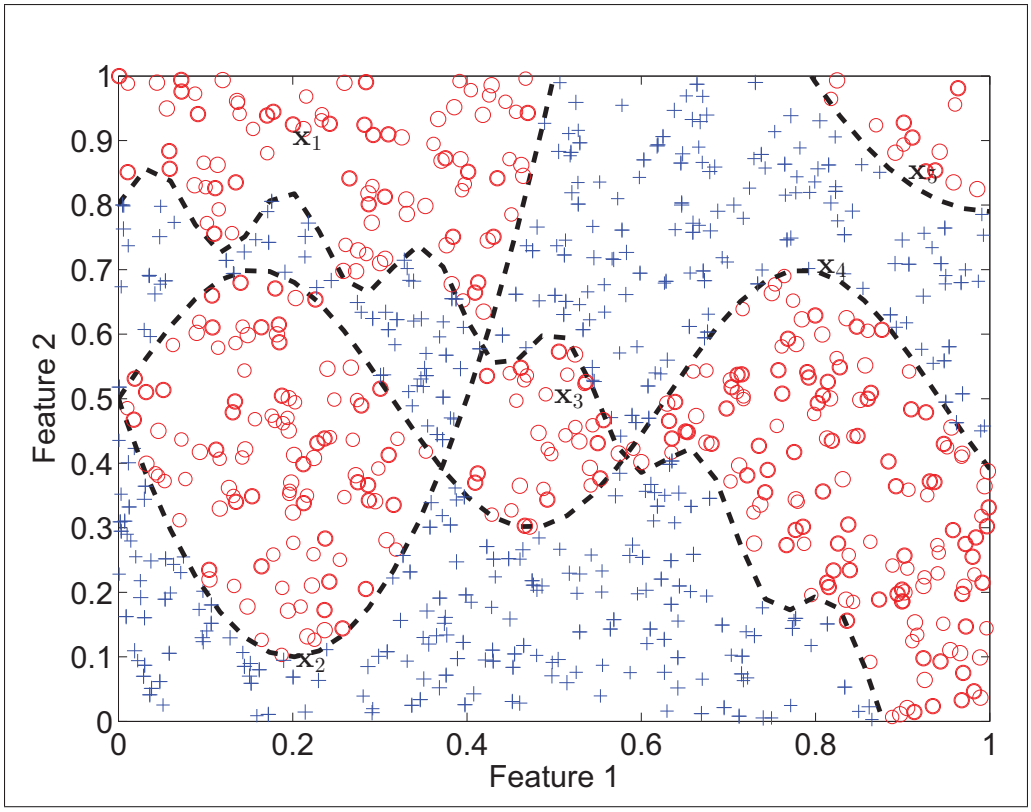


Figure 3.10 The dynamic selection dataset (DSEL) that is used to extract the meta-features. The set DSEL was generated with 500 samples, 250 for each class

As in our previous papers [2; 36], we consider the size of the region of competence $K = 7$, i.e., the seven nearest neighbors of the query sample, and the size of the output profiles set $K_p = 5$. Figure 3.11 shows the regions of competence of each training sample. The samples belonging to the region of competence θ_j, defined using DSEL, are shown for each testing sample separately (Figures 3.11 (b) to Figure 3.11 (f)).

For each test sample $\mathbf{x}_j$, five meta-feature vectors are extracted, each one corresponding to the behavior of one base classifier (c_1 to c_5) for the classification of $\mathbf{x}_j$. Tables 3.1 to 3.5 present

the meta-feature vectors obtained for each test sample and base classifier. For each instance $\mathbf{x}_j$, we present the meta-feature vectors computed for each of the 5 base classifiers as well as the decision obtained by the meta-classifier, denoted by $\delta_{i,j}$. $\delta_{i,j} = 1$ means that the base classifier was considered competent, and was thus used to predict the label of the query sample.

For the sample $\mathbf{x}_1$ (Table 3.1), it is an easier classification case since it is located close to the mean of one of the class centers (w_1). We can see in Figure 3.11 (b) that all instances in the region of competence of $\mathbf{x}_1$ belong to the same class. The classifiers c_1, c_3 and c_4 achieve a 100% recognition rate in the local region (as can be seen in Figure 3.7). This also holds true for the decision space, where those base classifiers present the correct label for the most similar output profiles as well. Thus it is clear that they are competent for the classification of $\mathbf{x}_1$.

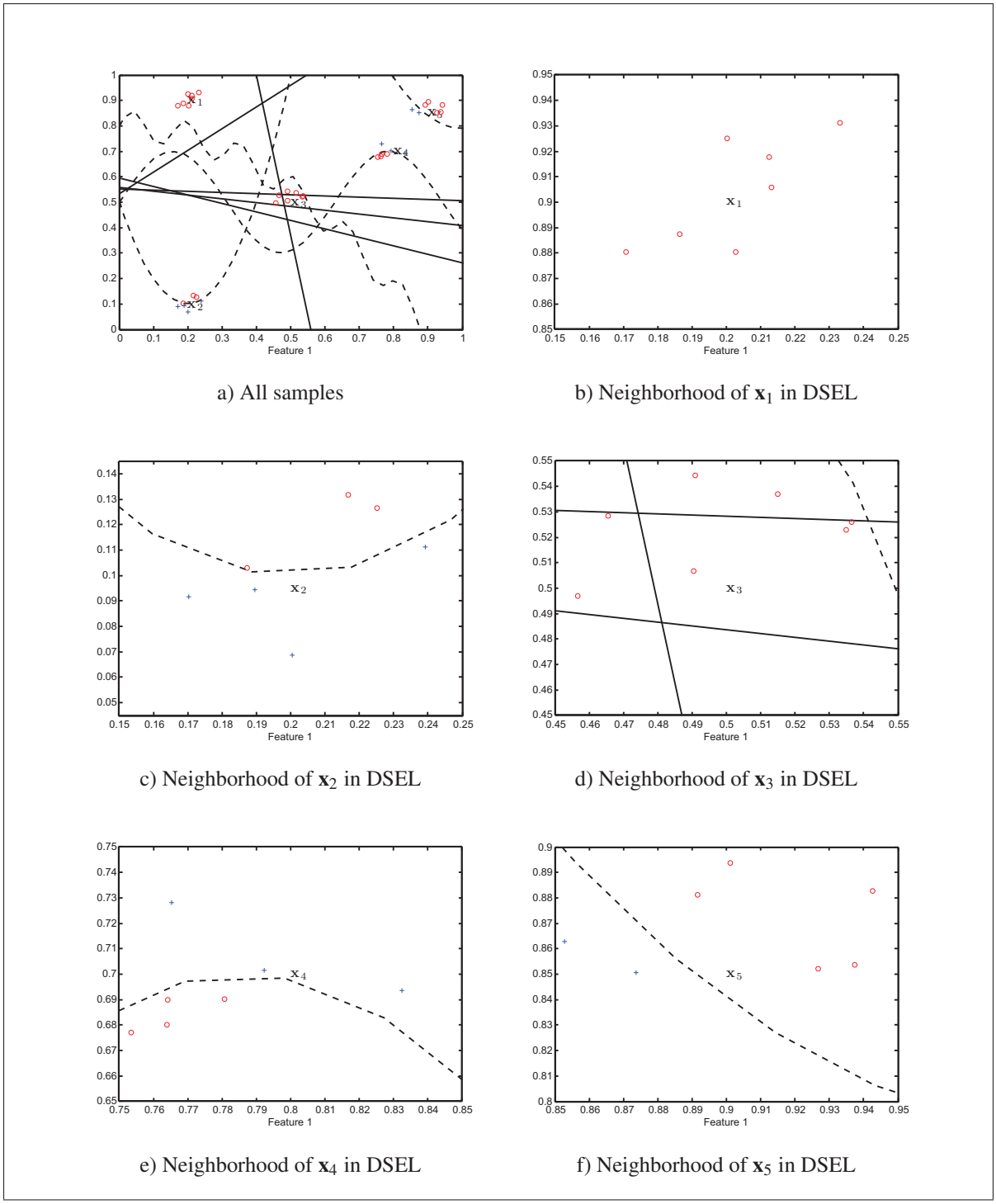
Table 3.1 Meta-Features extracted for the sample $\mathbf{x}_1$

	f_1							f_2							f_3	f_4					f_5	$\delta_{i,j}$
c_1	1	1	1	1	1	1	1	0.65	0.66	0.59	0.62	0.66	0.76	0.61	1.00	1	1	1	1	1	0.97	1
c_2	0	0	0	0	0	0	0	0.39	0.38	0.31	0.34	0.38	0.35	0.06	0.00	0	0	0	0	0	0.87	0
c_3	1	1	1	1	1	1	1	0.84	0.81	0.77	0.81	0.82	0.91	0.82	1.00	1	1	1	1	1	0.99	1
c_4	1	1	1	1	1	1	1	0.79	0.78	0.73	0.76	0.79	0.88	0.77	1.00	1	1	1	1	1	0.98	1
c_5	0	0	0	0	0	0	0	0.30	0.32	0.24	0.24	0.29	0.37	0.23	0.00	0	0	0	0	0	0.87	0

For the classification of the instance $\mathbf{x}_2$, we can see that it is located closer to the border separating the two classes. We can see that there are samples in the region of competence of $\mathbf{x}_2$ belonging to both classes. The base classifiers that achieve a good performance considering both the validation samples in the region of competence θ_j and the most similar output profiles, meta-feature f_4, are considered competent.

Table 3.2 Meta-Features extracted for the sample $\mathbf{x}_2$

	F1							F2							F3	F4					F5	$\delta_{i,j}$
c_1	1	0	1	1	0	0	1	1.00	0.00	0.97	1.00	0.00	0.00	0.89	0.57	1	0	1	1	1	1.00	1
c_2	1	0	1	1	0	0	1	0.67	0.32	0.63	0.62	0.33	0.39	0.59	0.57	1	0	1	1	1	0.97	1
c_3	1	0	1	1	0	0	1	0.89	0.11	0.87	0.87	0.14	0.17	0.81	0.57	1	0	1	1	1	0.99	1
c_4	1	0	1	1	0	0	1	0.86	0.13	0.83	0.85	0.15	0.18	0.81	0.57	1	0	1	1	1	0.99	1
c_5	0	1	0	0	1	1	0	0.28	0.70	0.19	0.20	0.67	0.79	0.23	0.43	0	1	0	0	0	0.87	0



a) All samples

b) Neighborhood of $\mathbf{x}_1$ in DSEL

c) Neighborhood of $\mathbf{x}_2$ in DSEL

d) Neighborhood of $\mathbf{x}_3$ in DSEL

e) Neighborhood of $\mathbf{x}_4$ in DSEL

f) Neighborhood of $\mathbf{x}_5$ in DSEL

Figure 3.11 Local regions computed using the K-Nearest Neighbor algorithm in the feature space. The region of competence of each testing sample is shown in one sub-figure

The sample $\mathbf{x}_3$ is located in a region close to the lines generated by the Perceptrons c_2, c_3, c_4 and c_5. However, all neighbor samples of $\mathbf{x}_3$ belong to the same class. Thus, the classifiers that achieve a good performance in the region of competence θ_j, and also for the set ϕ_j with the most the similar outputs profiles of $\tilde{\mathbf{x}}_3$, are selected. It is important to note that, in contrast to the testing instances $\mathbf{x}_1$ and $\mathbf{x}_2$, we can see that both the posterior probability meta-feature, meta-feature f_2, and the classifier's confidence, meta-feature f_5, produce lower results than the ones presented in Tables 3.1 and 3.2 since the samples are closer to the decision boundary of the base classifiers.

Table 3.3 Meta-Features extracted for the sample $\mathbf{x}_3$

	F1							F2							F3	F4					F5	$\delta_{i,j}$
c_1	0	0	0	0	0	0	0	0.12	0.13	0.00	0.06	0.15	0.13	0.08	0.00	0	0	0	0	1	0.39	0
c_2	1	1	1	0	0	1	1	0.60	0.51	0.46	0.43	0.47	0.60	0.45	0.71	1	0	1	1	0	0.66	1
c_3	1	1	1	1	1	1	1	0.56	0.60	0.53	0.49	0.56	0.68	0.54	1.00	1	1	1	1	0	0.66	1
c_4	0	1	0	0	0	0	1	0.47	0.52	0.43	0.41	0.48	0.57	0.45	0.29	0	0	0	0	0	0.36	0
c_5	0	0	0	0	0	0	0	0.48	0.47	0.40	0.43	0.46	0.44	0.41	0.00	0	0	0	0	1	0.36	0

For the sample $\mathbf{x}_4$ (Table 3.4), we can see that the majority of its neighbor samples come from a different class (Figure 3.11 (d)). If we consider dynamic selection techniques that are based solely on accuracy information, such as local classifier accuracy (LCA) [22] or overall classifier accuracy (OLA) [22], as well as the *a priori* and a posteriori methods [44], the base classifiers c_2, c_3 and c_4 are considered the most competent. So, using only the accuracy information in the local regions (region of competence) may not be sufficient to select the competent classifiers. However, these three classifiers predict the wrong label for $\mathbf{x}_4$; as shown in Figure 3.7, they would predict that $\mathbf{x}_4$ belongs to class 1 (red circle).

Table 3.4 Meta-Features extracted for the sample $\mathbf{x}_4$

	F1							F2							F3	F4					F5	$\delta_{i,j}$
c_1	1	0	1	0	0	1	0	0.92	0.37	0.93	0.00	0.37	1.00	0.00	0.43	1	1	1	0	0	0.99	1
c_2	0	1	0	1	1	0	1	0.42	0.66	0.22	0.60	0.63	0.39	0.59	0.57	0	0	0	1	1	0.90	0
c_3	0	1	0	1	1	0	1	0.16	0.81	0.06	0.77	0.77	0.17	0.75	0.57	0	0	0	1	1	0.90	0
c_4	0	1	0	1	1	0	1	0.34	0.64	0.27	0.59	0.61	0.37	0.58	0.57	0	0	0	1	1	0.90	0
c_5	1	0	1	0	0	1	0	0.62	0.37	0.57	0.31	0.37	0.72	0.32	0.43	1	1	1	0	0	0.89	0

Through the use of different meta-features, the META-DES is able to select a competent classifier (c_1) for the sample $\mathbf{x}_4$. The base classifier c_1 achieves a better performance in the decision space, (meta-feature f_4) (it is able to predict the correct class label for the closest samples in the decision space). Since each output profile $\tilde{\mathbf{x}}_k$ in the decision space is associated with a sample $\mathbf{x}_k$ in the feature space, we present the most similar output profiles of the sample $\tilde{\mathbf{x}}_4$. We can see that computing the similarity using the decision space yields distinct results, i.e., different validation samples are selected for extracting the meta-features. In this case, the closest output profiles, selected in the decision space, are from samples that belong to the same class of $\mathbf{x}_4$. So, the meta-features extracted using those samples are more likely to reflect the behavior of the base classifier c_1 for the classification of the sample $\mathbf{x}_4$. In addition, the base classifier c_1 also presents a higher posterior probability for the correct class label (meta-feature f_2), and a higher confidence in its answer for the classification of the query sample $\mathbf{x}_4$ (meta-feature f_5) when compared to the other base classifiers. Thus, it is considered as a competent classifier for the classification of the sample $\mathbf{x}_4$.

It is important to mention that the base classifier c_5 also predicts the correct label for the sample $\mathbf{x}_4$. However, it was not considered as a competent classifier since it presented lower confidence in its prediction (meta-feature f_5) as well as lower results for f_2 when compared to c_1.

Table 3.5 Meta-Features extracted for the sample $\mathbf{x}_5$

	F1							F2							F3	F4					F5	$\delta_{i,j}$
c_1	1	0	0	0	0	1	0	0.85	0.10	0.05	0.01	0.14	0.94	0.04	0.29	0	1	0	1	0	0.99	0
c_2	0	1	1	1	1	0	1	0.27	0.74	0.68	0.70	0.71	0.33	0.71	0.71	1	0	1	0	1	0.98	1
c_3	0	1	1	1	1	0	1	0.00	1.00	1.00	0.98	1.00	0.06	1.00	0.71	1	0	1	0	1	1.00	1
c_4	0	1	1	1	1	0	1	0.21	0.78	0.76	0.74	0.79	0.24	0.76	0.71	1	0	1	0	1	0.98	1
c_5	1	0	0	0	0	1	0	0.70	0.28	0.18	0.21	0.25	0.81	0.20	0.29	0	1	0	1	0	0.97	0

Considering these five testing samples, an interesting fact we can obtain from this example is the influence of using the decision space for estimating the competence of the base classifiers, especially considering the closest output profile (which holds the first position in the vector f_4). Based on Tables 3.1 to 3.5, when the base classifier predicts the correct label for the closest

(first) output profile of the query sample, the probability of the base classifier being selected as competent is high.

Figure 3.12 illustrates the decision boundary obtained by the META-DES framework. Using only five linear weak classifiers and dynamic selection, we can approximate the complex decision boundary of the P2 problem. The methodology used to define the decision boundary obtained by the technique is presented in 3.7.1.

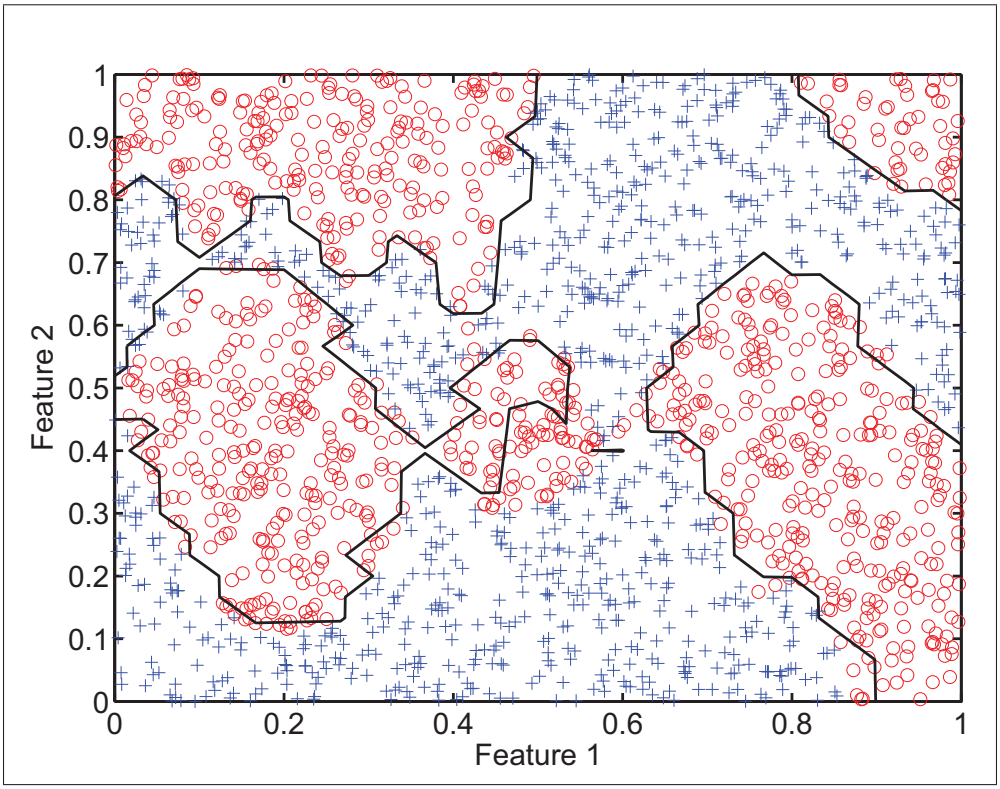


Figure 3.12 Decision Boundary obtained by the META-DES system using a pool of 5 Perceptrons. The META-DES achieves a recognition rate of 95.50% using 5 Perceptrons

When we apply static combination rules such as majority voting or Adaboost, the classification accuracy is much lower. Figure 3.13 illustrates the decision boundary obtained by static ensemble techniques using five Perceptron classifiers. We show the decisions obtained using the Average, Majority voting, Product, Maximum, as well as the Adaboost techniques. The av-

erage and product rules achieve a recognition rate of 47.5%, while the maximum and majority voting rules obtain an accuracy of 50%, and AdaBoost 56%. This can be explained by the fact that all classifiers in the pool are used to predict the label. However, due to the complexity of the problem, the degree of disagreement between the classifiers is very high. For the majority of test samples, half of the base classifiers disagree with the other half (predicts a different class label). The decisions of classifiers that are not experts for the local region end up negatively influencing the final decision. Thus, the static combination rule yields results that are close to random guessing. Even using techniques that assign weights to the base classifiers, such as Adaboost, we cannot approximate the complex decision of the P2 problem using only five linear classifiers.

3.5 Further Analysis

In this section, we evaluate the following aspects of the META-DES framework using the P2 problem:

a. The effect of the pool size on the classification accuracy.

b. The effect of the size of the dynamic selection dataset (DSEL) on the classification performance of the system.

c. The results of static the combination techniques for the P2 problem. This analysis is performed in order to provide an insight into why dynamic selection should be preferred for solving complex classification problems.

d. The results of classical pattern recognition techniques such as Support Vector Machines and Random Forest for the P2 problem.

For the sake of simplicity, we use the same methodology used in the previous section: 500 samples for training ($\mathcal{T}$), 500 instances for the meta-training dataset ($\mathcal{T}_\lambda$), 500 instances for the dynamic selection dataset *DSEL*, and 2000 samples for testing, $\mathcal{G}$. For each set, the prior

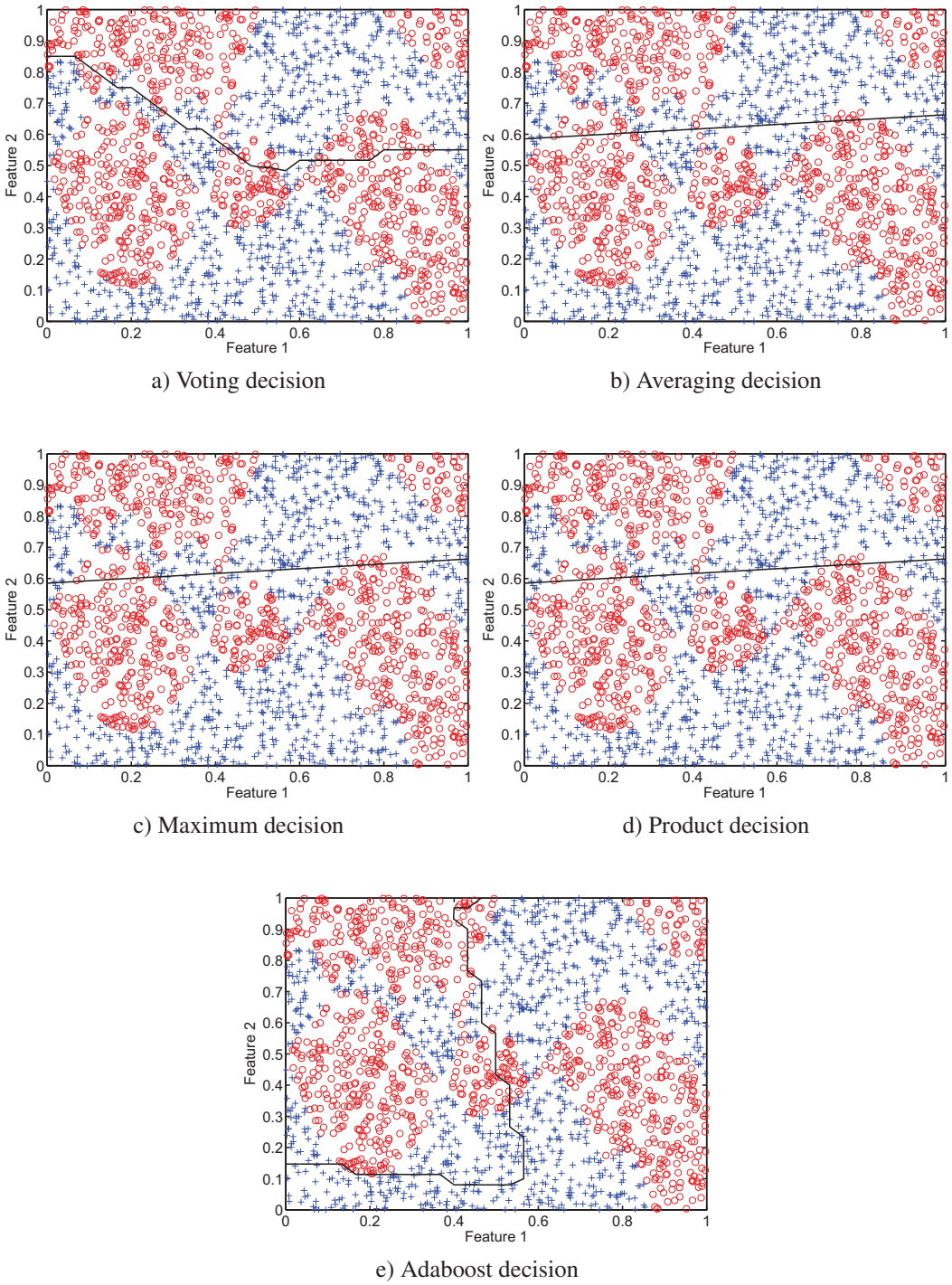


Figure 3.13 Decision boundaries generated by each static combination method. The pool of classifiers is composed of the 5 Perceptrons presented in Figure 3.6

probabilities of both classes are equal. Moreover, since the objective of this work is to study whether dynamic selection of linear classifier can solve complex non-linear classification problems, we also consider Decision Stumps [70] as base classifiers. We show that the META-DES framework works equally well using a pool of Decision Stumps.

3.5.1 The Effect of the Pool Size

For this experiment, we varied the size of the pool from 5 to 100 at 5 point intervals (20 results are obtained). The size of the dynamic selection dataset (DSEL) was set at 500 (as shown in Figure 3.10). The effect of the size of the pool of classifiers, M, is shown in Figure 3.14. We can see that the size of the pool does not have a significant impact on the classification accuracy of the META-DES, especially when the Perceptron is considered as the base classifier. This finding can be explained by the fact that using only 5 base classifiers, the Oracle (ideal selection scheme) achieves a classification accuracy of 99.5% and 100% using Perceptrons and Decision Stumps, respectively. In other words, using five base classifiers, it is possible to represent the whole feature space. The key to having good classification performance lies in defining a criterion to select the best classifier(s) for any given test sample. An interesting point is that the performance using decision stumps decreases as more classifiers are added to the pool, with the recognition performance decreasing when more than 25 base classifiers are used. Therefore, adding more classifiers does not always lead to higher classification accuracy.

Figures 3.15 and 3.16 illustrate the decision boundary obtained by the META-DES framework using Perceptron and Decision, respectively, stump as base classifier. We can see that when only 5 base classifiers are used, the decision boundary of the META-DES is close to the real decisions of the problem.

3.5.2 The effect of the size of the dynamic selection dataset (DSEL)

Figure 3.17 shows the performance of the META-DES using both Perceptron and Decision stumps according to the DSEL size. We varied the size of the dynamic selection dataset from

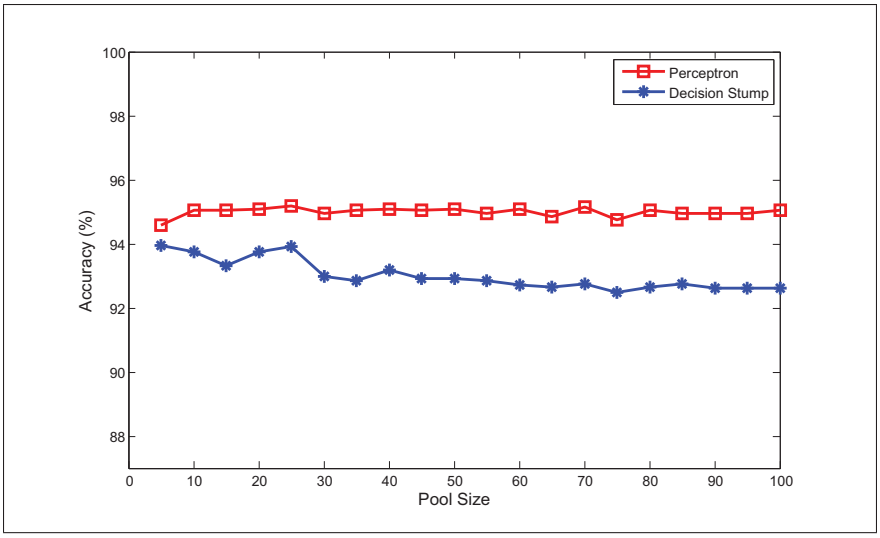


Figure 3.14 The effect of the pool size, M in the classification accuracy. Perceptron and Decision Stumps are considered as base classifiers

50 to 1000 at 50 point intervals (20 configurations were tested). The distribution varying the size of DSEL is presented in 3.7.4. For this experiment, the size of the pool was set at 100. We can observe that the size of the dynamic selection dataset has a greater influence on the classification result. This can be explained by the fact that the dynamic selection dataset, DSEL, is used in estimating the competence of the base classifiers, as shown in the classification example (Section 3.4.4). With more samples in DSEL, the probability of selecting samples that are similar to the query sample both in the feature space or in the decision space for extracting the meta-features is higher. Hence, a better estimation of the competence of the base classifiers is achieved.

Moreover, to better understand the influence of both the size of the pool and the size of the dynamic selection dataset together, we constructed a 3D mesh plot showing the accuracy of the system according to both parameters (Figures 3.18 and 3.19).

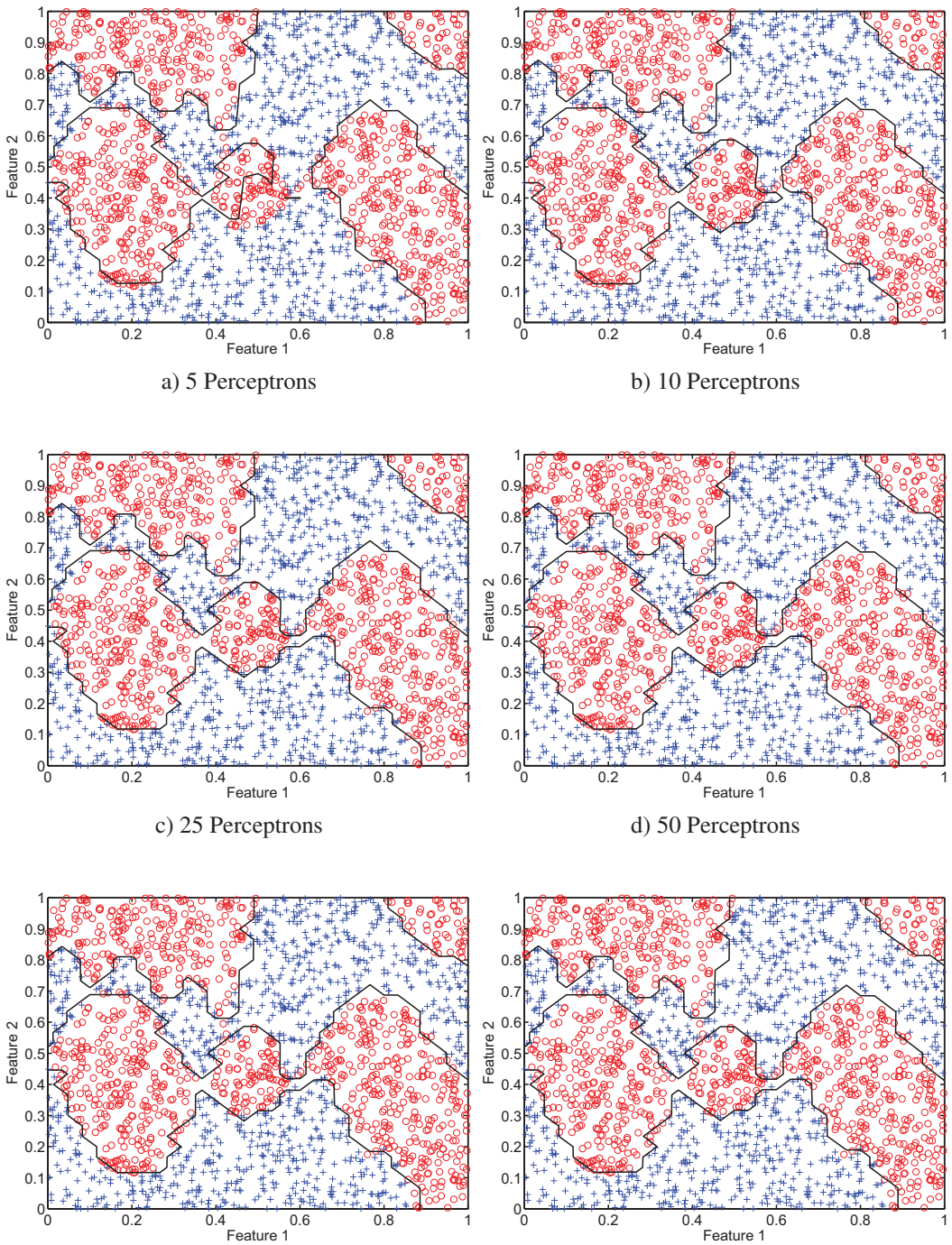


Figure 3.15 Decision boundaries generated by the META-DES framework for different pool size. Perceptrons are used as base classifiers

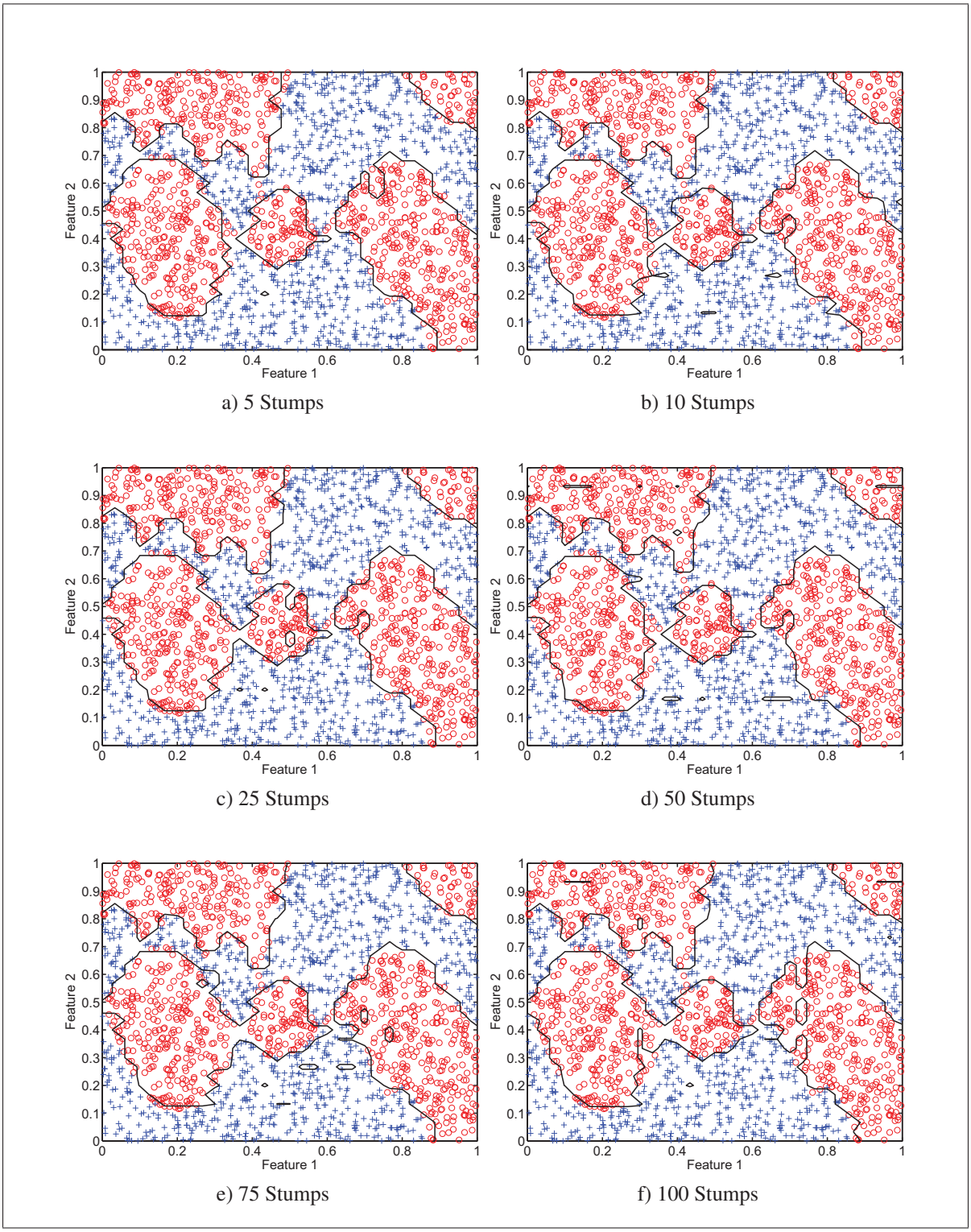


Figure 3.16 Decision boundaries generated by the META-DES framework for different pool size. Decision Stumps are used as base classifiers

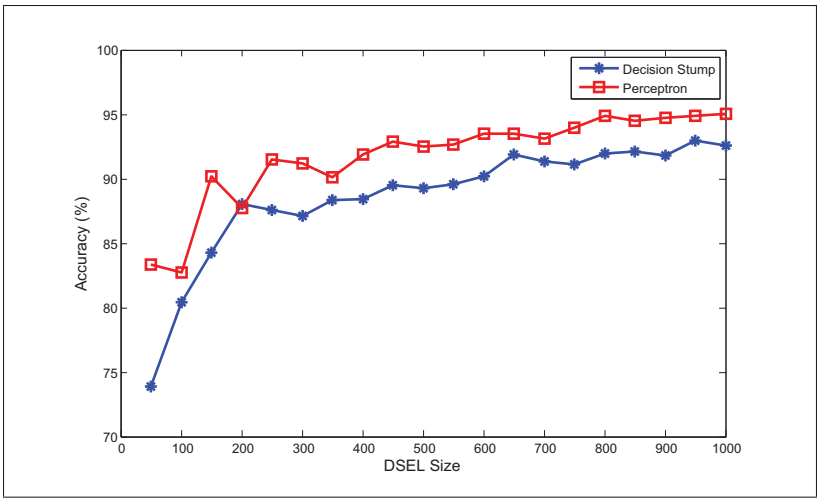


Figure 3.17 The effect of the DSEL size in the classification accuracy. Perceptron and Decision Stumps are considered as base classifiers. The results are obtained using a pool with 100 base classifiers, $M = 100$

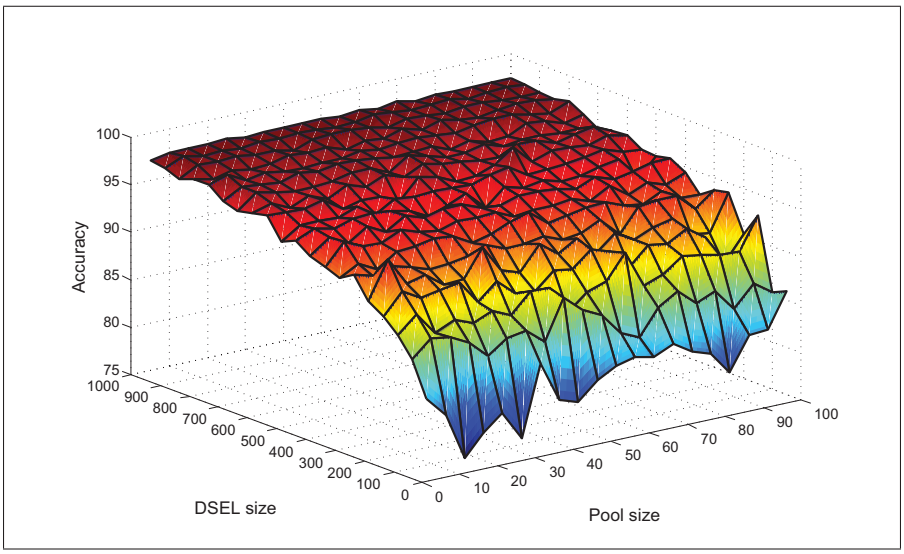


Figure 3.18 The effect of the pool size and the validation set size (DSEL) in the accuracy of the system. Perceptrons are used as base classifier

3.5.3 Results of static combination techniques

Figures 3.20 and 3.21 illustrate the accuracy rates of static combination techniques by varying the size of the pool of classifiers. Furthermore, the decision boundaries for the static combi-

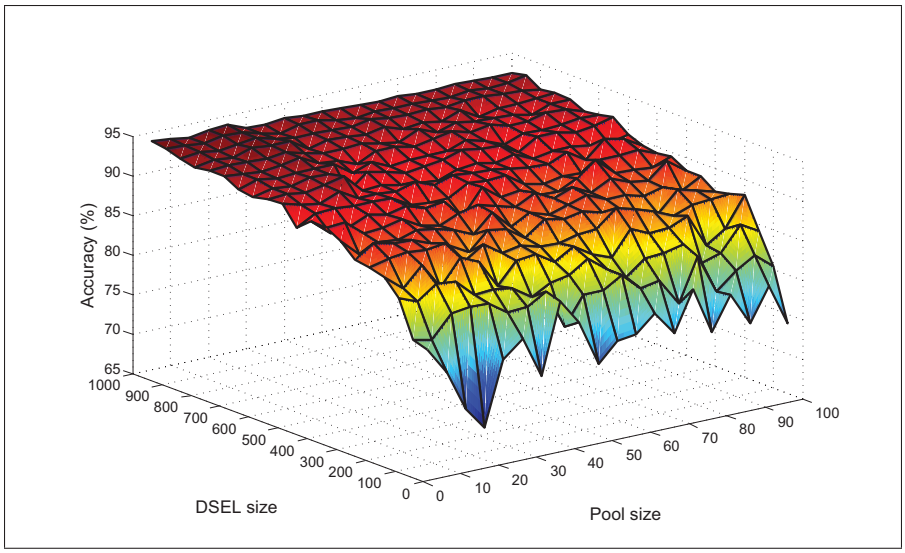


Figure 3.19 The effect of the pool size and the validation set size (DSEL) in the accuracy of the system. Decision Stumps are used as base classifier

nation techniques are shown in Figures 3.22 and 3.23 for Perceptrons and Decision Stumps, respectively.

Even when the size of the pool is increased to 100 base classifiers (Figure 3.22), the static combination techniques cannot approximate the decision of the P2 problem. The performance using Decision Stumps as base classifiers is significantly better than that using Perceptrons for the static combination rules, especially considering the AdaBoost technique. This fact can be explained by the divide-and-conquer approach of decision stumps, in which each Stump is trained using a single feature. Hence, the classification task may become easier for the classifier model. However, the classification accuracy is still far from the performance obtained by the META-DES framework. Even using only 5 base classifiers, the performance of the META-DES is superior when compared to static combination techniques using up to 100 base classifiers.

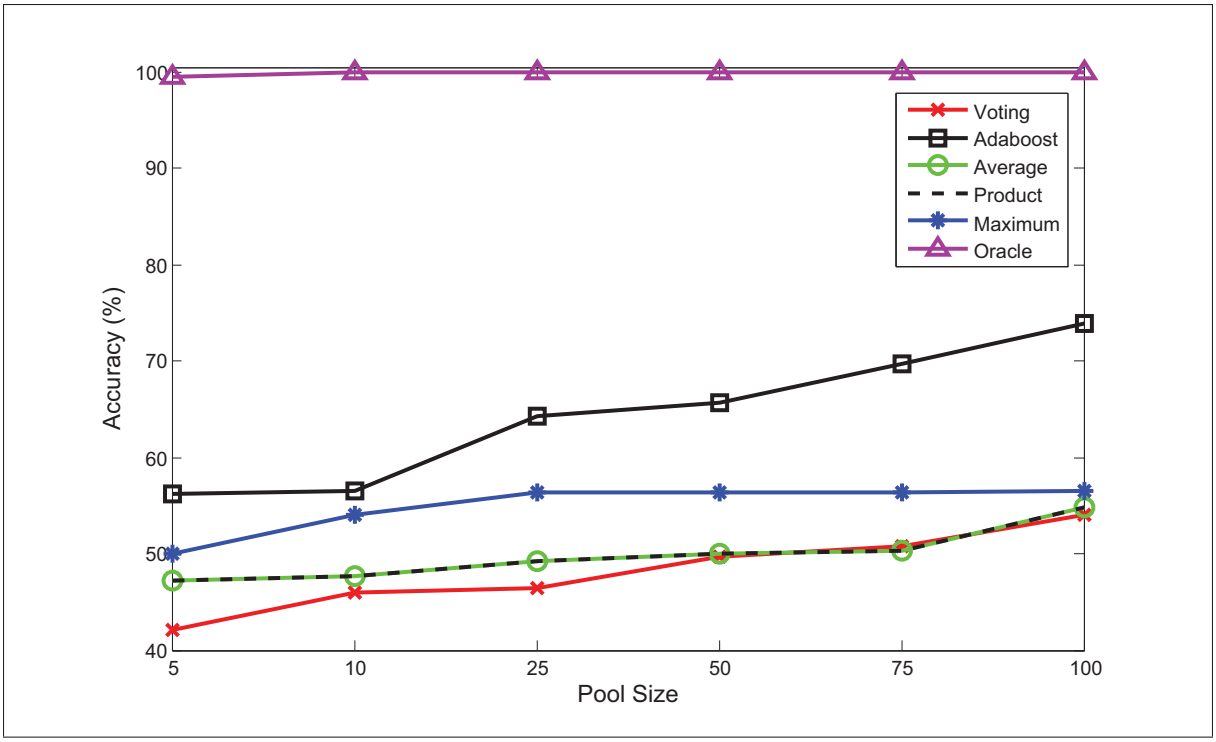


Figure 3.20 Results of static combination techniques using Perceptron as base classifier

3.5.4 Single classifier models

In this section, we show the results of classical classification models for the P2 problem. We evaluate three classifier models: MLP Neural Network, Support Vector Machines with Gaussian Kernel (SVM) and Random Forest classifier. These classifiers were selected based on a recent study [71] that ranked the best classification models in a comparison considering a total of 179 classifiers over 121 classification datasets. All the classifiers were evaluated using the Matlab PRTOOLS toolbox [63]. The parameters of each classifier were set as follows:

a. MLP Neural Network LM: The validation data was used to select the number of nodes in the hidden layer. We used a configuration with 100 neurons in the hidden layer. The training process was performed using the Levenberg-Marquadt algorithm. The training process was stopped if its performance on the validation set decreased or failed to improve for five consecutive epochs.

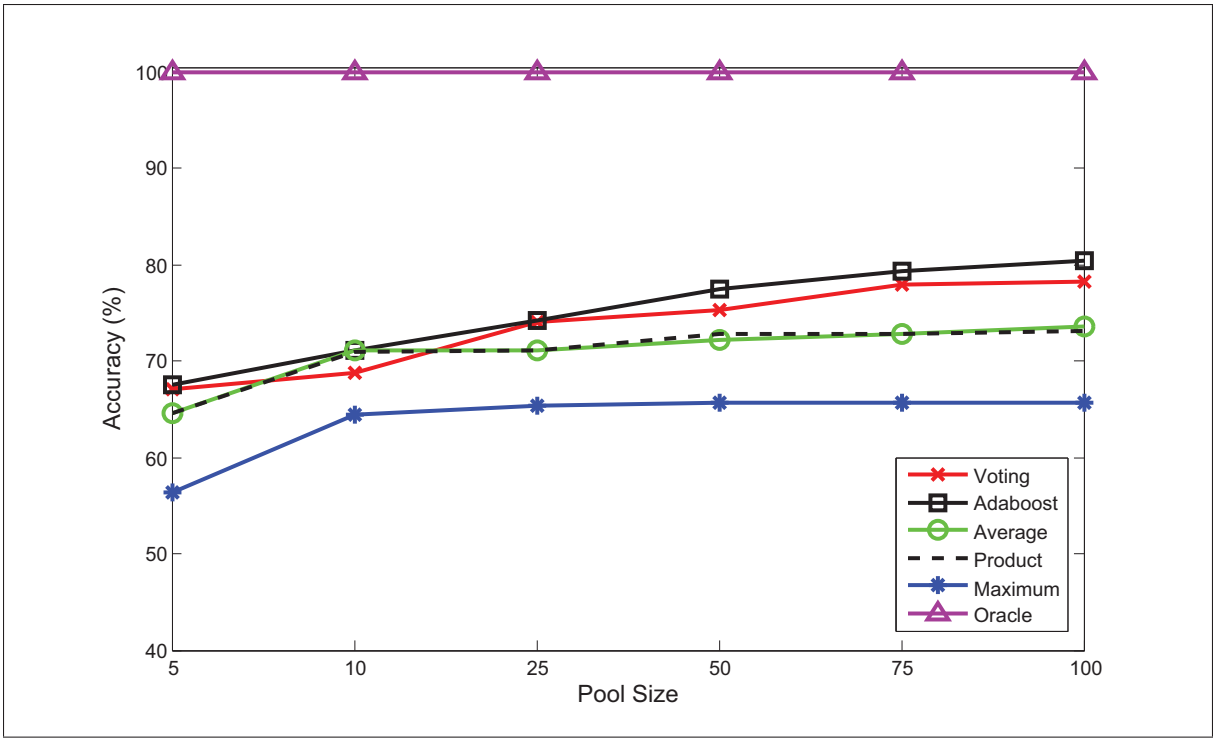


Figure 3.21 Results of static combination techniques using Decision Stumps as base classifier

b. MLP Neural Network RPROP: The validation data was used to select the number of nodes in the hidden layer. We used a configuration with 100 neurons in the hidden layer. The training process was performed using the Resilient Backpropagation algorithm [72] since this algorithm presented both a faster convergence and better classification performance in many applications [11]. The training process was stopped if its performance on the validation set decreased or failed to improve for five consecutive epochs.

c. SVM: A radial basis SVM with a Gaussian Kernel was used. A grid search was performed in order to set the values of the regularization parameter c and the Kernel spread parameter γ.

d. Random Forest: We vary the number of trees from 25 to 200 at 25 point intervals. The configuration with the highest performance on the validation dataset is used for generalization. Since there are only two features in the P2 problem, a decision stump is used (depth = 1).

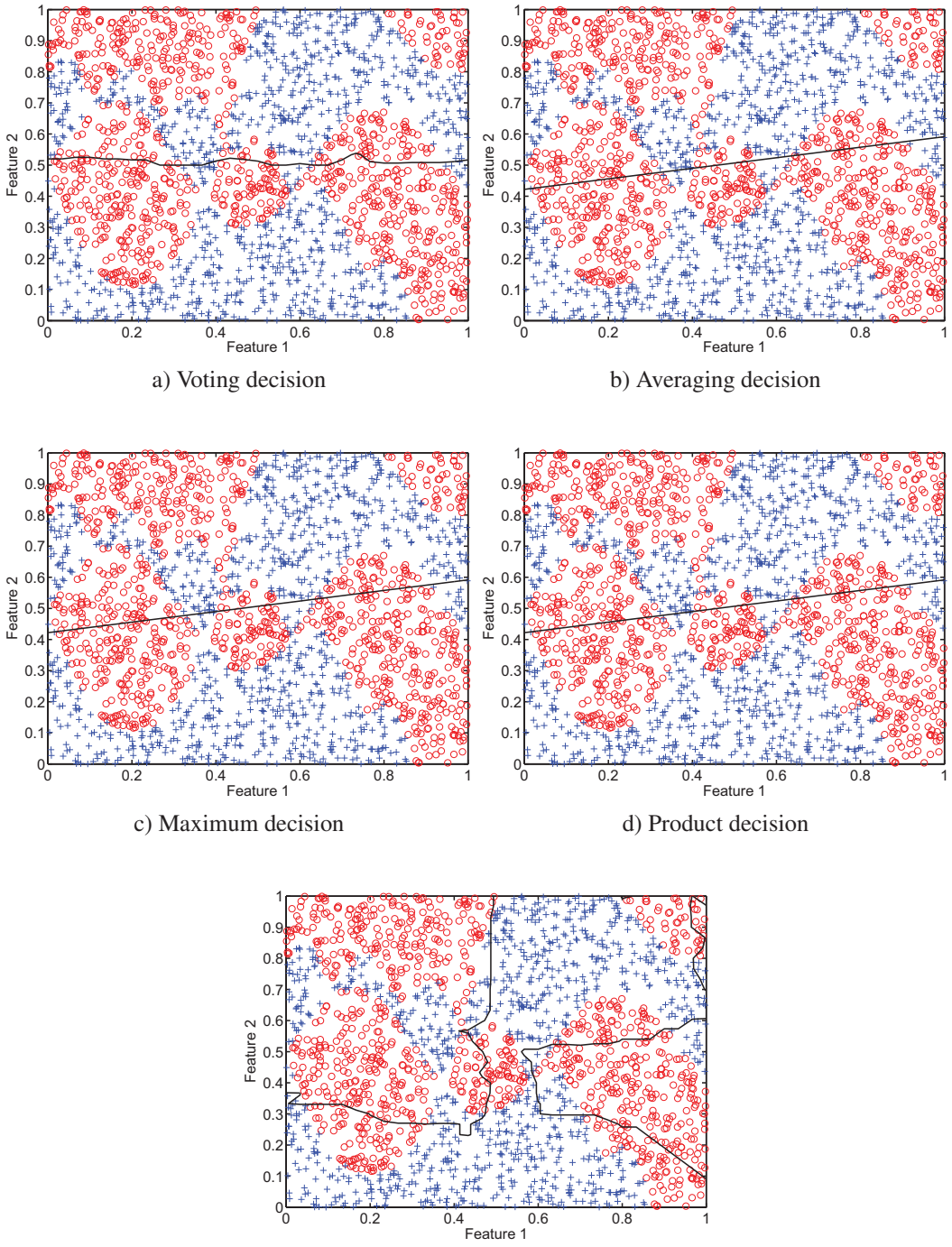
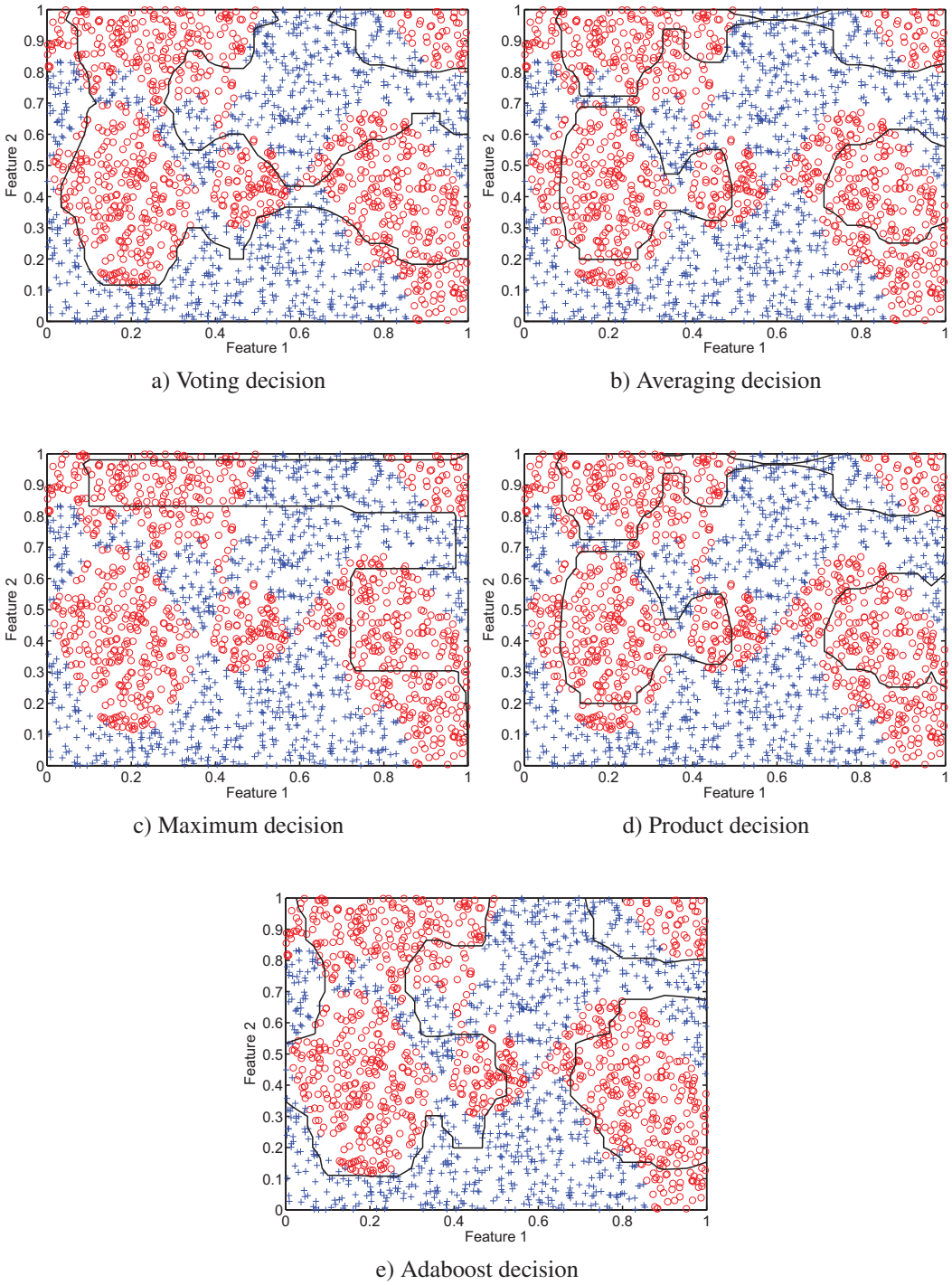


Figure 3.22 Decision boundaries generated by each ensemble method. The pool of classifiers is composed of 100 Perceptron classifiers



a) Voting decision

b) Averaging decision

c) Maximum decision

d) Product decision

e) Adaboost decision

Figure 3.23 Decision boundaries generated by each ensemble method. The pool of classifiers is composed of 100 Decision stumps classifiers

Since these classifiers do not require a meta-training stage, in these experiments, we merge the training ($\mathcal{T}$) and meta-training set ($\mathcal{T}_\lambda$) into a single training set, thereby training the classifiers with 1000 samples. The samples in the dynamic selection dataset (DSEL) are used for the validation dataset. The decision boundary obtained by each classifier is presented in Figure 3.24. The MLP neural network trained with Levemberg-Marquadt obtained a recognition accuracy of 90%, while that trained with Resilient Backpropagation algorithm obtained 77%. The SVM obtained a recognition accuracy of 93%, and the random forest classifier achieved 91%. The classification accuracy of these single classifier models is lower than the performance of the META-DES using a pool of either five Perceptrons or five Decision Stumps. This result can be explained by the complex nature of the P2 problem. It is difficult to properly train a strong classifier to learn the separation between the two classes. These classifiers might require more training samples in order to obtain better generalization performance.

3.6 Conclusion

In this chapter, we perform a DEEP analysis of the META-DES framework using linear classifiers. The analysis is conducted using the P2 problem, which is a complex non-linear problem with two classes having multiple class centers. We demonstrate that using the META-DES framework, we can approximate the complex non-linear distribution of the P2 problem using few linear classifiers. The accuracy rate provided by the best linear classifiers trained for this problem is around 50%. We demonstrate that using static combination techniques, it is impossible to approximate the complex decision frontier of the P2 problem. Because of the complex nature of the P2 problem, for every test sample, there is high disagreement between the predictions made by the base classifier. Since there is no consensus regarding the correct label for the test sample, the static combination techniques end up making random decisions. Even using techniques that assign weights to the base classifiers, such as AdaBoost, the classification accuracy using 100 base classifiers is still very different from the performance of the META-DES framework. Classifiers that are not experts in the local region where the query instance is located end up negatively influencing the decision of the system. Using dynamic selection, the

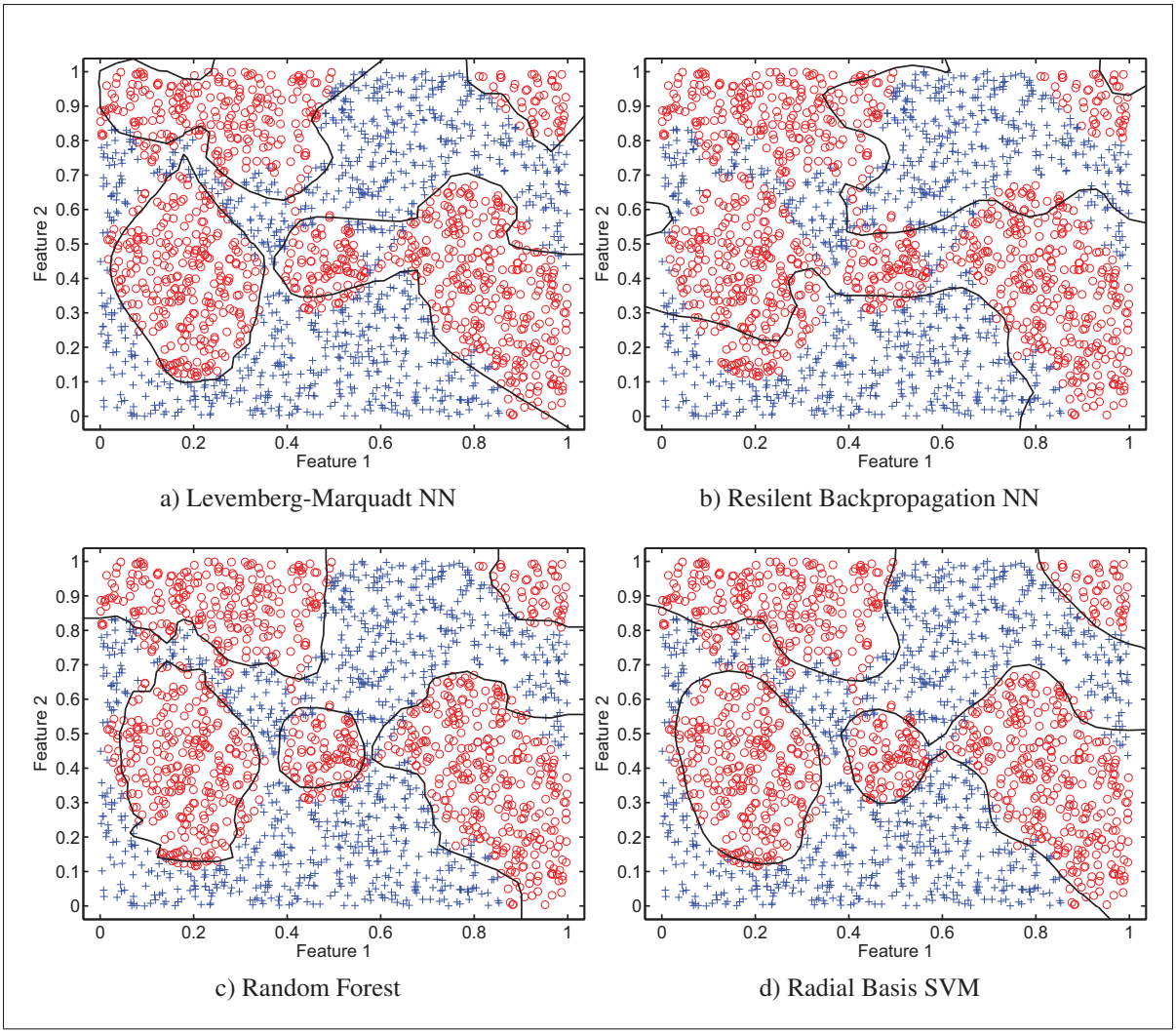


Figure 3.24 Decision boundaries obtained using a single classifier. (a) MLP-NN with 100 neurons in the hidden layer trained using Levemberg-Marquadt (90% accuracy). (b) MLP-NN with 100 neurons in the hidden layer trained using Resilient Backpropagation (77% accuracy). (c) Random Forest classifier (91% accuracy). Support Vector Machine with a Gaussian kernel (d) (93% accuracy)

decisions of the base classifiers that are not experts for the given query sample are not taken into account. Only the most competent classifiers are selected to the predict the label of the query sample.

The size of the pool of classifiers did not have a significant influence on the recognition rate. This finding can be explained by the fact that using only 5 base classifiers, the Oracle perfor-

mance of the Pool is at 100%. In other words, there is at least one base classifier that predicts the correct class label for every testing sample. The crucial element here is the criteria used to estimate the level of competence of the base classifiers in order to always select those that predict the correct class label for a given test sample. Moreover, we noticed a performance drop when using decision stumps as base classifiers when more than 25 base classifiers are used. These results indicate that increasing the number of base classifiers in the pool does not always lead to greater classification accuracy. Thus, one aspect of the framework that must be further investigated is how many base classifiers should be trained in the overproduction phase for a given classification problem.

We evaluate the impact of the pool of classifiers and the size of the dynamic selection dataset (DSEL) that is used in dynamically estimating the level of competence of the base classifier. Experimental results show that the size of the dynamic selection dataset has a higher impact on classification performance. This can be explained by the fact the majority of the meta-features proposed for the META-DES framework are extracted from instances in DSEL that are similar to the query sample, considering both the feature space and the decision space. With more samples in the DSEL, the probability of selecting samples that are similar to the query sample in both the feature space and in the decision space for extracting the meta-features is higher. Hence, a better estimation of the competence of the base classifiers is achieved. The results found in this analysis should be considered as a guideline for future work on the META-DES and for other dynamic ensemble selection based on local accuracy information in general.

Furthermore, the META-DES framework presented a higher classification accuracy for the P2 problem than did the classical single classifier model. This finding may be attributed to the complex nature of the P2 problem, since a classifier such as an SVM or an MLP neural network may require more training samples for a better generalization performance. Using dynamic selection through the META-DES framework we can approximate the complex decision of the P2 problem using less training data.

It is important to mention that there is still room for improvement in the META-DES framework. Using five base classifiers, the accuracy rate obtained by the META-DES is around 95%, while the Oracle performance is close to 100%. Future works will involve the definition of new meta-features in order to achieve a behavior that is closer to the ideal dynamic selection technique (Oracle).

3.7 Appendix

3.7.1 Plotting decision boundaries

When dealing with dynamic classifier or ensemble selection, for each classification sample $\mathbf{x}_{j,test}$, a specific ensemble or base classifier is selected to perform the classification. Thus, a grid is generated over the 2D image. The grid is generated in the same interval as the P2 classification problem [0, 1] for both axes. Each point on the 2D grid is passed down to the dynamic selection technique in order to predict its label. After every point on the 2D grid is evaluated, the MATLAB contour plot is used to separate the points that were classified between the two classes. It is important to mention that the number of points on the grid influences the definition of the decision boundary. A high number of points in the grid leads to a more precise decision boundary. In our experiment, we use a 100×100 grid, for a total of 10,000 points, in order to have a more precise decision boundary map. For the static combination rules and classification models the decision boundaries are plotted using the *plotc* function from the PRTOOLS Matlab Toolbox [63].

3.7.2 Ensemble Generation

Figures 3.25 and 3.26 illustrate the pool of classifiers generated with bagging using Perceptrons and Decision Stumps, respectively. We consider a pool of 5, 10, 25, 50, 75 and 100 base classifiers. Considering a pool size of 100 classifiers, we can see that most of the classifiers are in the same region. Thus, we believe the majority of classifiers are redundant. This can be explained by the fact we used bagging for the generation of the pool. In the bagging technique,

the bootstraps are randomly taken from the training data, and such, there is no guarantee that a high diversity pool will be achieved. The use of techniques such as the Random Oracle [6], may be considered in the future as an alternative for the generation of the pool in order to achieve higher diversity at the pool level.

3.7.3 Sample Selection Mechanism: consensus threshold h_c

In this section, we show the results of the sample selection mechanism by varying the value of the threshold h_c. Since the sample selection mechanism depends on the base classifier (i.e., the consensus among the pool), we show the result of the sample selection mechanism using both Perceptrons and Decision Stumps Figures 3.27 and 3.28 respectively.

Samples close to the decision boundary are the ones more likely to be selected for the training of the meta-classifier. Hence, the sample selection mechanism focuses on samples that are close to the decision boundaries thus, are harder to predict its correct label. This principle is similar to the support vectors in the SVM, where samples close to the decision boundaries are used to achieve the best separating hyperplanes. In our case, the samples close to the decision boundary are used to train the meta-classifier in order to distinguish between a competent classifier from an incompetent one in cases where a disagreement exists between the base classifiers in the pool.

3.7.4 Size of the dynamic selection dataset (DSEL)

Figure 3.25 shows the dynamic selection dataset (DSEL) generated with different sizes. The figures show the exact distributions of the dataset DSEL used to evaluate the performance of the META-DES framework according to its size (Section 3.5.2). The size of the DSEL has a significant impact on the performance of the META-DES framework 3.17. This can be explained by the fact the meta-features are extracted based on the neighborhood of the query sample $\mathbf{x}_{j,test}$ projected in DSEL.

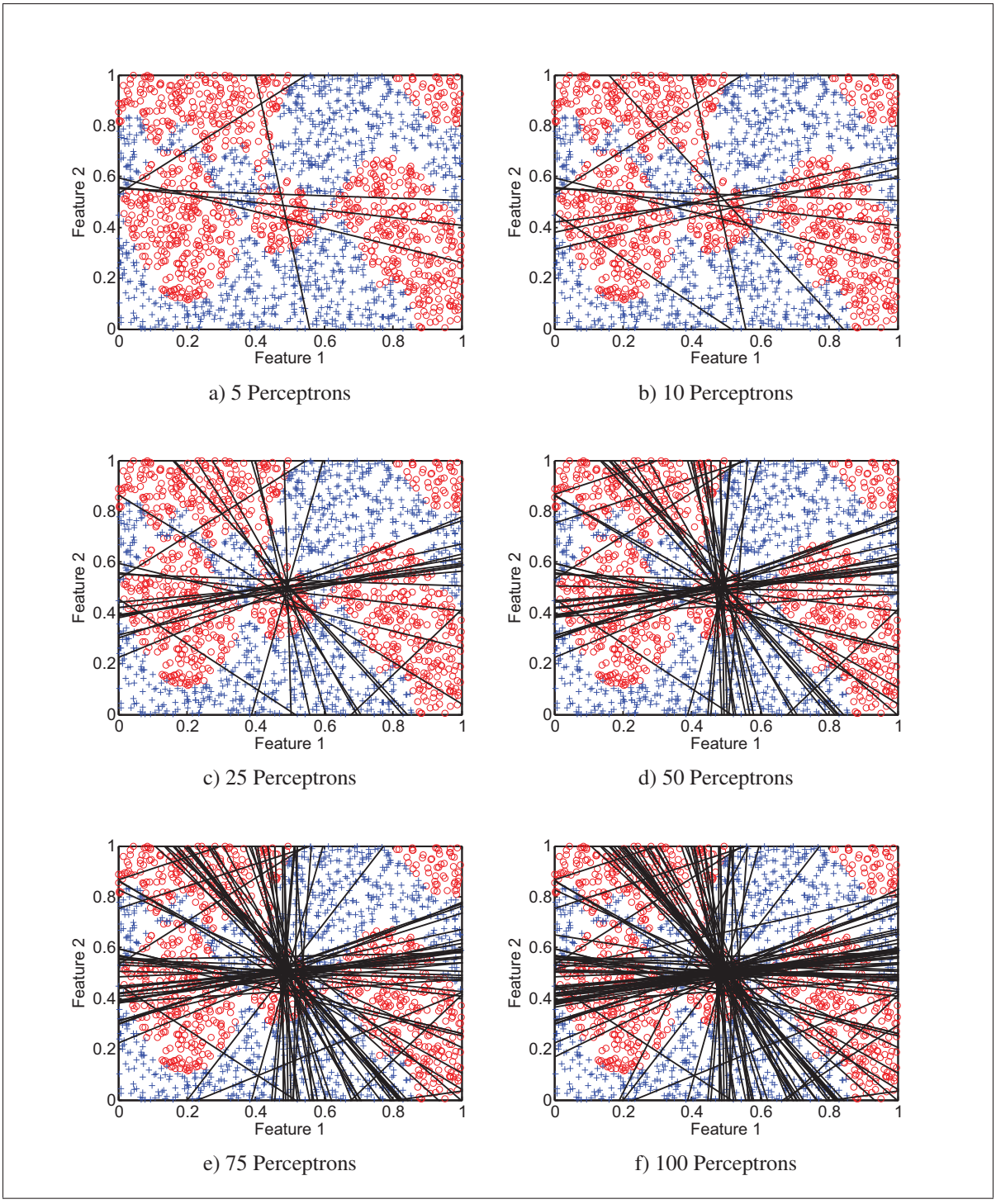
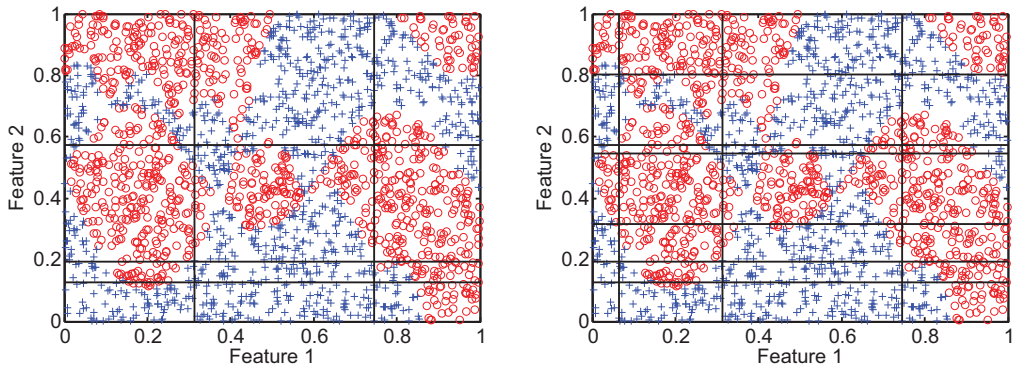
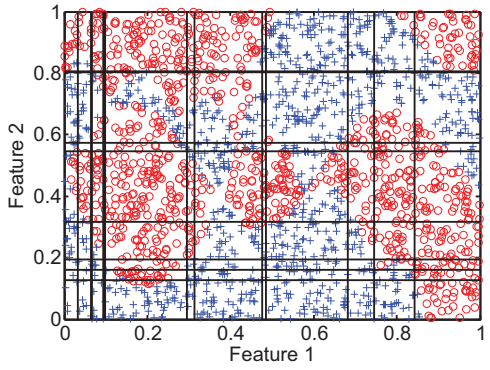


Figure 3.25 Base classifiers generated during the overproduction phase. The Bagging technique is used to generate the pool of classifiers

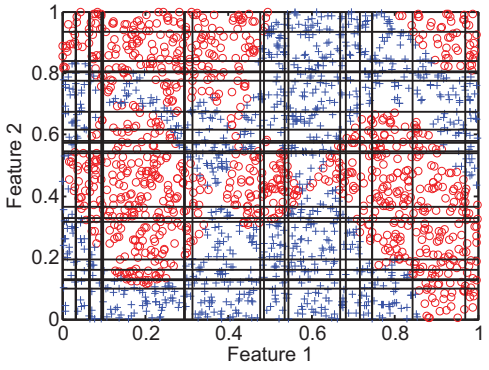


a) 5 Stumps

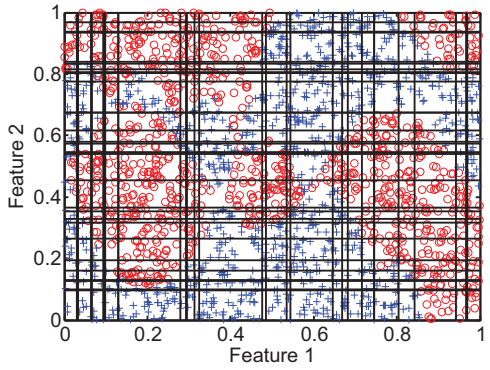
b) 10 Stumps



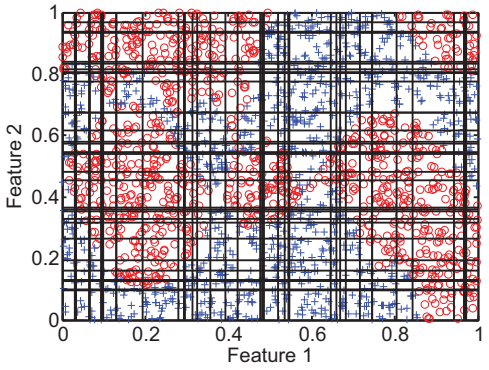
c) 25 Stumps



d) 50 Stumps

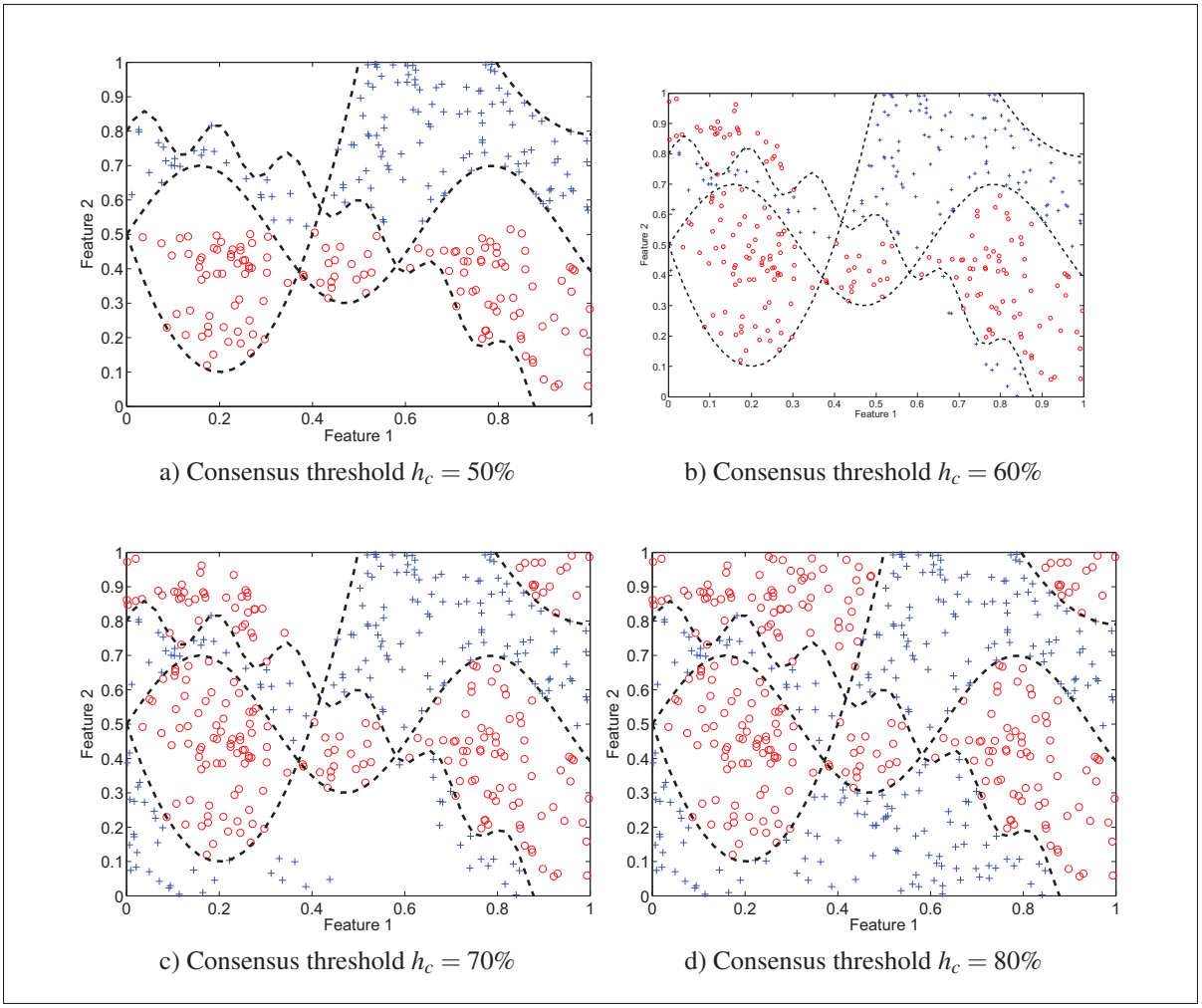


e) 75 Stumps



f) 100 Stumps

Figure 3.26 Decision Stumps classifiers generated during the overproduction phase. The Bagging technique is used to generate the pool of classifiers



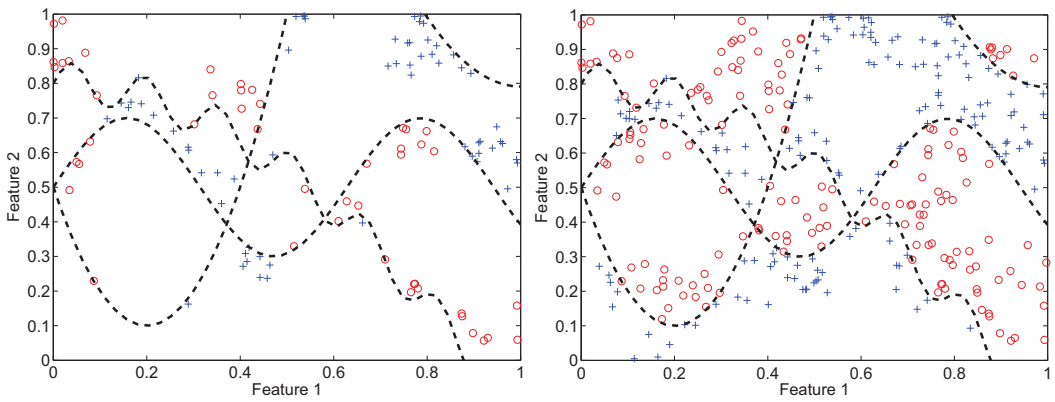
a) Consensus threshold $h_c = 50\%$

b) Consensus threshold $h_c = 60\%$

c) Consensus threshold $h_c = 70\%$

d) Consensus threshold $h_c = 80\%$

Figure 3.27 Meta-training dataset $\mathcal{T}_\lambda$ after the sample selection mechanism is applied. A pool composed of 100 Perceptrons is used



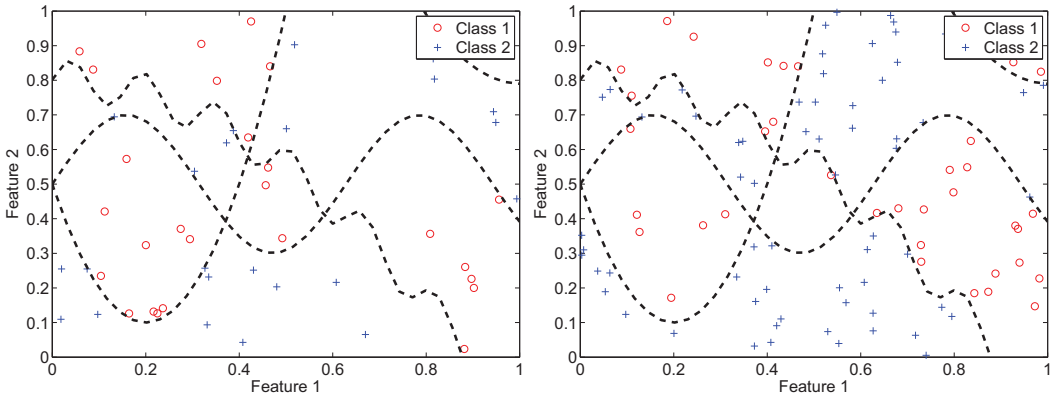
a) Consensus threshold $h_c = 50\%$

b) Consensus threshold $h_c = 50\%$

c) Consensus threshold $h_c = 50\%$

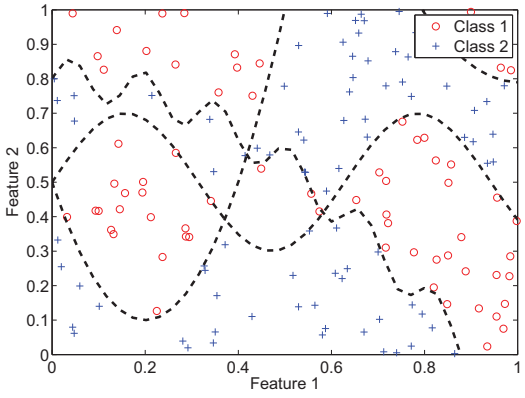
d) Consensus threshold $h_c = 50\%$

Figure 3.28 Meta-training dataset $\mathcal{T}_\lambda$ after the sample selection mechanism is applied. A pool composed of 100 Decision Stumps is used

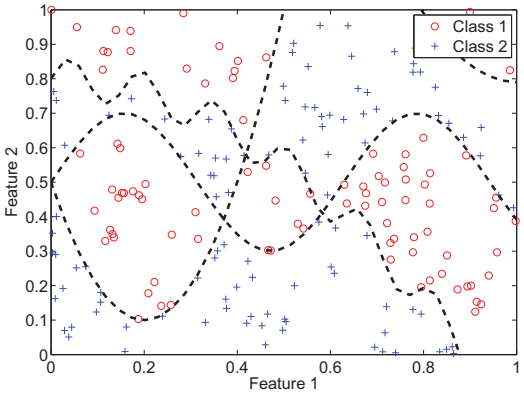


a) DSEL 50 Samples

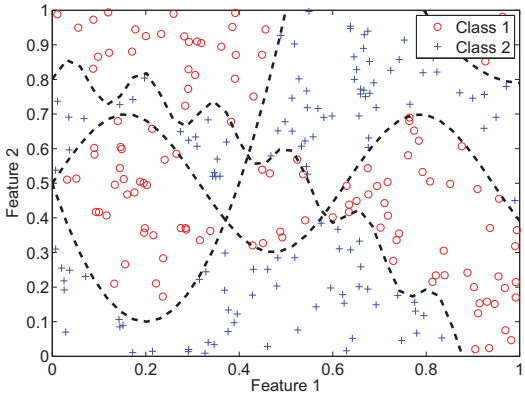
b) DSEL 100 Samples



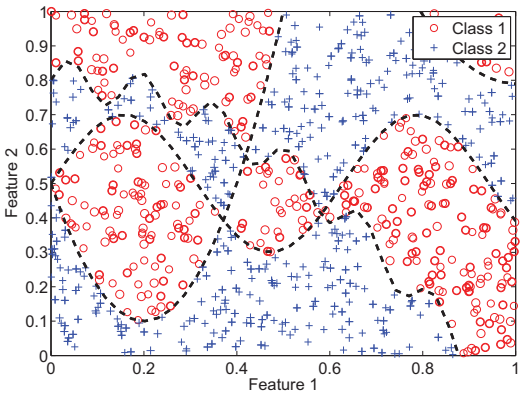
c) DSEL 150 Samples



d) DSEL 200 Samples



e) DSEL 250 Samples



f) DSEL 500 Samples

Figure 3.29 Distributions of the dynamic selection dataset (validation), used to extract the meta-features during the generalization phase of the system

CHAPTER 4

META-DES.ORACLE: META-LEARNING AND FEATURE SELECTION FOR DYNAMIC ENSEMBLE SELECTION

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Submitted to « Information Fusion » 2015.

Abstract

Dynamic ensemble selection (DES) techniques work by estimating the competence level of each classifier from a pool of classifiers, and selecting only the most competent ones for the classification of a specific test sample. The key issue in DES is defining a suitable criterion for calculating the classifiers' competence. There are several criteria available to measure the level of competence of base classifiers, such as local accuracy estimates and ranking. However, using only one criterion may lead to a poor estimation of the classifier's competence. In order to deal with this issue, we have proposed a novel dynamic ensemble selection framework using meta-learning, called META-DES. A meta-classifier is trained, based on the meta-features extracted from the training data, to estimate the level of competence of a classifier for the classification of a given query sample. An important aspect of the META-DES framework is that multiple criteria can be embedded in the system encoded as different sets of meta-features. However, some DES criteria are not suitable for every classification problem. For instance, local accuracy estimates may produce poor results when there is a high degree of overlap between the classes. Moreover, a higher classification accuracy can be obtained if the performance of the meta-classifier is optimized for the corresponding data. In this paper, we propose a novel version of the META-DES framework based on the formal definition of the Oracle, called META-DES.Oracle. The Oracle is an abstract method that represents an

ideal classifier selection scheme. A meta-feature selection scheme using an overfitting cautious Binary Particle Swarm Optimization (BPSO) is proposed for improving the performance of the meta-classifier. The difference between the outputs obtained by the meta-classifier and those presented by the Oracle is minimized. Thus, the meta-classifier is expected to obtain results that are similar to the Oracle. Experiments carried out using 30 classification problems demonstrate that the optimization procedure based on the Oracle definition leads to a significant improvement in classification accuracy when compared to previous versions of the META-DES framework and other state-of-the-art DES techniques.

4.1 Introduction

Multiple Classifier Systems (MCS) aim to combine classifiers in order to increase the recognition accuracy in pattern recognition systems [24; 9]. MCS are composed of three phases [1]: (1) Generation, (2) Selection, and (3) Integration. In the first phase, a pool of classifiers is generated. In the second, a single classifier or a subset having the best classifiers of the pool is(are) selected. We refer to the subset of classifiers as the Ensemble of Classifiers (EoC). In the last phase, called integration, the predictions of the selected classifiers are combined to obtain the final decision.

The classifier selection phase can be either static or dynamic. In static selection, the ensemble is selected during the training stage. The classifiers with the best performance, according to the selection criteria, considering the whole training or validation distribution are selected to compose the ensemble. Then, the ensemble is used for the classification of all unseen data. In dynamic approaches, the ensemble of classifiers is selected during the test phase. For each test sample, the competence of the base classifiers is estimated according to a selection criterion. Then, only the classifier(s) that attain a certain competence level, are used to predict the label of the given test sample. Recent works in the MCS literature have shown that dynamic ensemble selection (DES) techniques achieve higher classification accuracy when compared to static ones [1; 2; 14]. This is especially true for ill-defined problems, i.e., for problems where the size of the training data is small, and there are not enough data available to train the classifiers [16;

17]. Moreover, using dynamic ensemble selection, we can solve classification problems with a complex non-linear decision boundary using only a few linear classifiers, while static ensemble techniques, such as Bagging and AdaBoost, cannot [37].

When dealing with DES, the key issue is to define a suitable criterion to select the most competent classifiers to predict the label of a specific query sample. Several criteria have previously been proposed, based on different sources of information, such as the classifier local accuracy estimates in small regions of the feature space surrounding the query instance, called the region of competence [22; 14], probabilistic models [18; 45; 39], ranking [38] and classifier behavior [21; 16]. In our previous work [2], we proposed a novel DES framework using meta-learning, called META-DES. The framework is divided into three steps: (1) Overproduction, where the pool of classifiers is generated; (2) Meta-training, where the meta-features are extracted using the training data, and used as inputs to train a meta-classifier that works as a classifier selector; and (3) the Generalization phase, in which the meta-features are extracted from each query sample and used as input to the meta-classifier. The meta-classifier decides whether the base classifier is competent enough to classify the test sample. The main advantage of the META-DES framework is its modularity. Any criterion used to estimate the level of competence of base classifiers can be encoded as a new set of meta-features and added to the system. A total of five sets of meta-features were proposed in [2], each one representing a different DES criterion, such as local accuracy information and degree of confidence. Moreover, in [37], a case study is presented demonstrating how the use of multiple criteria leads to a more robust dynamic selection technique. Using multiple sets of meta-features, even though one criterion might fail due to imprecisions in the local regions of the feature space or due to low confidence results, the system can still achieve a good performance as other meta-features are considered by the selection scheme. Because the META-DES framework considers the dynamic selection problem as a meta-classification problem, we can significantly improve the recognition accuracy of the system by focusing only on optimizing the performance of the meta-classifier.

However, there are some drawbacks to the META-DES framework. First, there are different sources of information that were not considered by the previous version of the system, such as probabilistic models, ambiguity, and ranking. Secondly, all sets of meta-features are used for every classification problem with no pre-processing step at all. As stated by the "No Free Lunch" theorem [33], there is no criterion for dynamic selection that outperforms all others over all possible classes of problems. Different classification problems may require distinct sets of meta-features. The meta-classifier training process is not optimized for each classification problem. This can also lead to low classification results, since we found that the training of the meta-classifier is problem-dependent [36]. For these reasons, the results obtained by the META-DES framework were still far from those achieved by the Oracle. The Oracle is an abstract model defined in [32], which always selects the classifier that predicted the correct label, for the given query sample, if such a classifier exists. The Oracle performance is used in order to know whether the performances achieved by DES techniques are close to the upper limit performance or whether there is still room for improvements in classification accuracy.

In this paper, ten new sets of meta-features are proposed, using sources of information that were not explored in the previous version framework, such as ranking, probabilistic models applied over the decisions obtained by the meta-classifier, and ambiguity, for a better estimation of the competence level of the base classifiers. The additional meta-features are motivated by a recent analysis conducted in [55], demonstrating that using different sources of information to estimate the competence level of the base classifiers is complementary and that combining them leads to a more robust DES technique.

In order to better address the behavior of the Oracle, we first provide a formal definition of the way the Oracle estimates the competence level of the base classifier. Following that, a meta-feature selection scheme using an overfitting cautious Binary Particle Swarm Optimization (BPSO) is conducted to optimize the performance of the meta-classifier, based on the Oracle definition. The difference between the level of competence estimated by the meta-classifier and that estimated by the Oracle is used as the fitness function for the BPSO. In other words, the BPSO seeks a meta-features vector that minimizes the difference between the behavior of the

meta-classifier and that of the Oracle in estimating the competence level of the base classifiers. Thus, the meta-classifier is more likely to present results that are closer to that of the Oracle. We call the proposed system META-DES.Oracle, since the formal definition of the Oracle is used during the training stage of the meta-classifier.

Lastly, the classification stage is performed using a hybrid dynamic selection and weighting scheme. The classifiers that attain a certain level of competence are selected to compose the ensemble. Next, the meta-classifier is used to compute the weights of the selected base classifiers to be used in a weighted majority voting scheme. Base classifiers that present a higher level of competence have greater influence on the classification of the query sample.

Experiments are conducted over 30 classification problems derived from different data repositories. We compare the results obtained by the proposed META-DES.Oracle with 10 state-of-the-art dynamic selection techniques, as well as static ensemble methods, such as AdaBoost [5]. Furthermore, we also compare the results obtained by the proposed META-DES.Oracle with those achieved by single classifier models, such as SVM with Gaussian Kernel and Random Forest. The goal of the experimental study is to answer the following research questions: (1) Are different sets of meta-features better suited for different problems? (2) Are all 15 sets of meta-feature relevant? (3) Does the META-DES.Oracle obtain a significant gain in classification accuracy when compared to the previous versions of the META-DES framework? (4) Does the META-DES.Oracle outperform state-of-the-art DES techniques? (5) Is the performance obtained by the proposed framework comparable to that of the best families of classifiers in the literature [71]?

This paper is organized as follows: Section 4.2 introduces state-of-the-art techniques for dynamic classifier and ensemble selection. The META-DES.Oracle is detailed in Section 4.3. In Section 4.4, we describe the 15 sets of meta-features proposed in this work. An illustrative example using synthetic data is shown in Section 4.5. The experimental study is conducted in Section 4.6. Finally, our conclusion and future works proposals are given in the last section.

4.2 Related Works

In this section, we present an overview of the approaches for dynamic ensemble selection and feature selection using evolutionary computation. They serve as complement to the motivations of this work.

4.2.1 Dynamic selection

Dynamic selection of classifiers consists of finding a single classifier c_i or an ensemble of classifiers C' that has the most competent classifiers to predict the label for a specific test sample, $\mathbf{x}_j$, based on a pool of classifiers C. This is a different concept from static selection, where the ensemble of classifiers C' is selected during the training phase, and considering the global performance of the base classifiers using either the training or validation dataset [10; 11; 12; 13].

The most important component of DES techniques is the criterion used to measure the level of competence of a base classifier c_i for the classification of a given query sample $\mathbf{x}_j$. The most common approach involves estimating the accuracy of the base classifiers in small regions of the feature space surrounding the query sample, $\mathbf{x}_j$, called the region of competence. This region is usually defined based on the KNN-rule applied to either the training [22] or validation data [14]. Based on the region of competence, there are several sources of information that can be used to measure the competence of the classifier in the DES literature [1]: Measures based solely on accuracy, such as the Overall Local Accuracy (OLA) [22], Local Classifier Accuracy (LCA) [22] and Modified Local Accuracy (MLA) [22], ranking information such as the Classifier Rank [38] and the simplified classifier rank [22], probabilistic information calculated over the decision obtained by the base classifiers such as the Kullback Leibler divergence, DES-KL [45] and the randomized reference classifier DES-PRC [18], classifier behavior calculated using output profiles such as the KNOP technique [16] and the KNORA family of techniques [14] using Oracle information. Furthermore, there are some selection criteria that estimate the competence level of a whole ensemble of classifiers rather than the competence of

each base classifier individually, such as the degree of consensus used in the Dynamic Over-production and Choose technique (DOCS) [15], diversity [42] and data handling [19].

An important concept in the DES literature is the definition of the Oracle. The Oracle is an abstract model defined in [32], which always selects the classifier that predicted the correct label, for the given query sample, if such a classifier exists. In other words, it represents the ideal classifier selection scheme. The Oracle is used in the DES literature in order to determine whether the results obtained by the proposed DES techniques is close to ideal accuracy or whether there is still room for improvements. As reported in a recent survey [1], the results obtained by DES techniques based solely on one source of information are still far from those achieved by the Oracle. As stated by Ko et al. [14], addressing the behavior of the Oracle is much more complex than applying a simple neighborhood approach, and the task of figuring out its behavior based merely on the pattern feature space is not an easy one. In addition, in our previous work [20], we demonstrated that the use of local accuracy estimates alone is insufficient to achieve good generalization performance.

To address these issues, in [2] we proposed a novel DES framework using meta-learning, called META-DES. From a meta-learning perspective, the dynamic selection problem can be seen as another classification problem, called the meta-problem. This meta-problem uses different criteria regarding the behavior of a base classifier in order to decide whether or not a base classifier c_i is competent enough to classify a given sample $\mathbf{x}_j$. In this paper, our aim therefore is to optimize the performance of the meta-classifier, using the meta-classification environment, to obtain results closer to those of the Oracle.

4.2.2 Feature selection using Binary Particle Swarm Optimization (BPSO)

Given a set of features m, the objective of feature selection is to identify the most informative subset of features $m^{'} \in m$. The reasons for using feature selection methods [73] are: removal of redundant and irrelevant features, reduction of dimensionality, reduction of the computational complexity of the system, as well as improvement of the classification accuracy. There are

two main factors when dealing with feature selection: the evaluation method, which is applied to compute the fitness of each solution, and the search strategy, which is used to explore the feature space in the search for a more suitable subset of features.

For the search strategy, the recent focus in the feature selection literature has been on evolutionary computation techniques, such as Genetic Algorithms (GA) [74; 75], Particle Swarm Optimization (PSO) [76; 77; 78], Differential Evolution (DE) [79; 80; 77] and Ant Colony Optimization (ACO) [81]. Evolutionary computation techniques have been shown to outperform other feature selection methods, such as sequential feature selection SFS, in many applications, especially when dealing with larger feature vectors, i.e., for classification problems with more than 50 features [82].

Particle Swarm Optimization (PSO) is an evolutionary computation technique inspired from the social behavior of birds flocking [76]. PSO is one of the most used evolutionary algorithms, due to its simplicity and low computational cost. The technique is based on a group of particles flying around in the search space to find the best solution. Recent works have shown the preference for PSO over other classical optimization techniques, such as GA because GA has too many parameters to set. Moreover, GA is very sensitive to the probability of crossover and mutation operators, as well as to the initial population of solutions. Therefore, it is likely to get stuck into local minima [73].

Since we are dealing with feature selection, a binary version of the PSO algorithm, BPSO is considered. BPSO has been shown in many applications to outperform other optimization algorithms in performing feature selection [76; 83; 84]. There are many versions of the BPSO algorithm, such as the Improved BPSO [83], CatfishBPSO [78] and MBPSO [85]. Current research the BPSO literature shows that the most important factor for achieving good convergence and avoiding local minima is the transfer function [77], that is responsible for mapping the continuous search space into a binary space. Generally speaking, there are two main types of transfer functions, S-shaped and V-shaped [77]. The main difference between the two families derives from the observation that the S-Shaped functions force the particles to switch 0 or

1 values at each generation, while the V-Shaped transfer functions encourage particles to stay in their current position when their velocity values are low, and switch the values only when the velocity is high. For these reasons, V-Shaped transfer functions were shown to be better both in terms of robustness to local minima and convergence speed. In this work, we consider one S-Shaped transfer function and one V-Shaped function, which presented the best overall performance, considering 25 benchmark functions [77].

4.3 The META-DES.Oracle

The META-DES framework is based on the assumption that the dynamic ensemble selection problem can be considered as a meta-problem [36]. This meta-problem uses different criteria regarding the behavior of a base classifier c_i, in order to decide whether it is competent enough to classify a given test sample $\mathbf{x}_j$. The meta-problem is defined as follows [2]:

- The **meta-classes** are either "competent" (1) or "incompetent" (0) to classify $\mathbf{x}_j$.
- Each set of **meta-features** f_i corresponds to a different criterion for measuring the level of competence of a base classifier.
- The meta-features are encoded into a **meta-features vector** $v_{i,j}$.
- A **meta-classifier** λ is trained based on the meta-features $v_{i,j}$ to predict whether or not c_i will achieve the correct prediction for $\mathbf{x}_j$, i.e., if it is competent enough to classify $\mathbf{x}_j$.

An overview of the META-DES framework is illustrated in Figure 4.1. The framework is divided into three phases: (1) Overproduction, (2) Meta-training, and (3) Generalization. Phases (1) and (2) are performed in offline mode, i. e., during the training stage of the framework. In the overproduction phase, the pool of classifiers C is generated using the training set $\mathcal{T}$. The following step is the meta-training stage, in which the meta-features are extracted for the training of the meta-classifier λ. In this stage, the meta-features are extracted from the meta-training set, $\mathcal{T}_\lambda$, and from the dynamic selection dataset, $DSEL$. The meta-data extracted from

$\mathcal{T}_\lambda$, denoted by $\mathcal{T}_\lambda^*$, are used for the training of the meta-classifier, and those extracted from *DSEL*, denoted by *DSEL**, are used as validation data during the BPSO optimization process. Phase (3) is conducted on-the-fly, with the arrival of each new test sample, $\mathbf{x}_{j,test}$, coming from the generalization dataset $\mathcal{G}$. For each base classifier c_i, a meta-features vector $v_{i,j}$ is extracted, corresponding to the behavior of the base classifier c_i for the classification of $\mathbf{x}_{j,test}$. $v_{i,j}$ is passed down to the meta-classifier λ that estimates if c_i is competent enough to predict the label for $\mathbf{x}_{j,test}$. After all the classifiers in the pool C are evaluated, the selected classifiers C' are combined using a weighted majority voting approach to predict the label w_l of $\mathbf{x}_{j,test}$. The main changes to the META-DES framework proposed in this paper are highlighted in different colors:

a. The meta-feature extraction process, in which 15 sets of meta-features are extracted. Ten new sets of meta-features are proposed in this work in order to explore different sources of information for estimating the competence level of the base classifiers, such as probabilistic models, ambiguity, behavior and ranking. The meta-feature extraction process is presented in Section 4.4.

b. The meta-features selection using Binary Particle Swarm Optimization and guided by Oracle information for achieving a behavior closer to the Oracle. The meta-features selection process is detailed in Section 4.3.2.2.

c. The combination approach, where a hybrid dynamic selection and weighting approach is considered for the classification of the query sample $\mathbf{x}_{j,test}$ (Section 4.3.3).

4.3.1 Overproduction

Similarly to [2], the Overproduction phase is performed using the Bagging technique [3]. The Bagging technique works by randomly selecting different bootstraps of the data for training each base classifier c_i. Each bootstrap uses of 50% of the training data. The pool of classifiers C is composed of 100 linear Perceptrons for the two-class problems and 100 multi-class linear

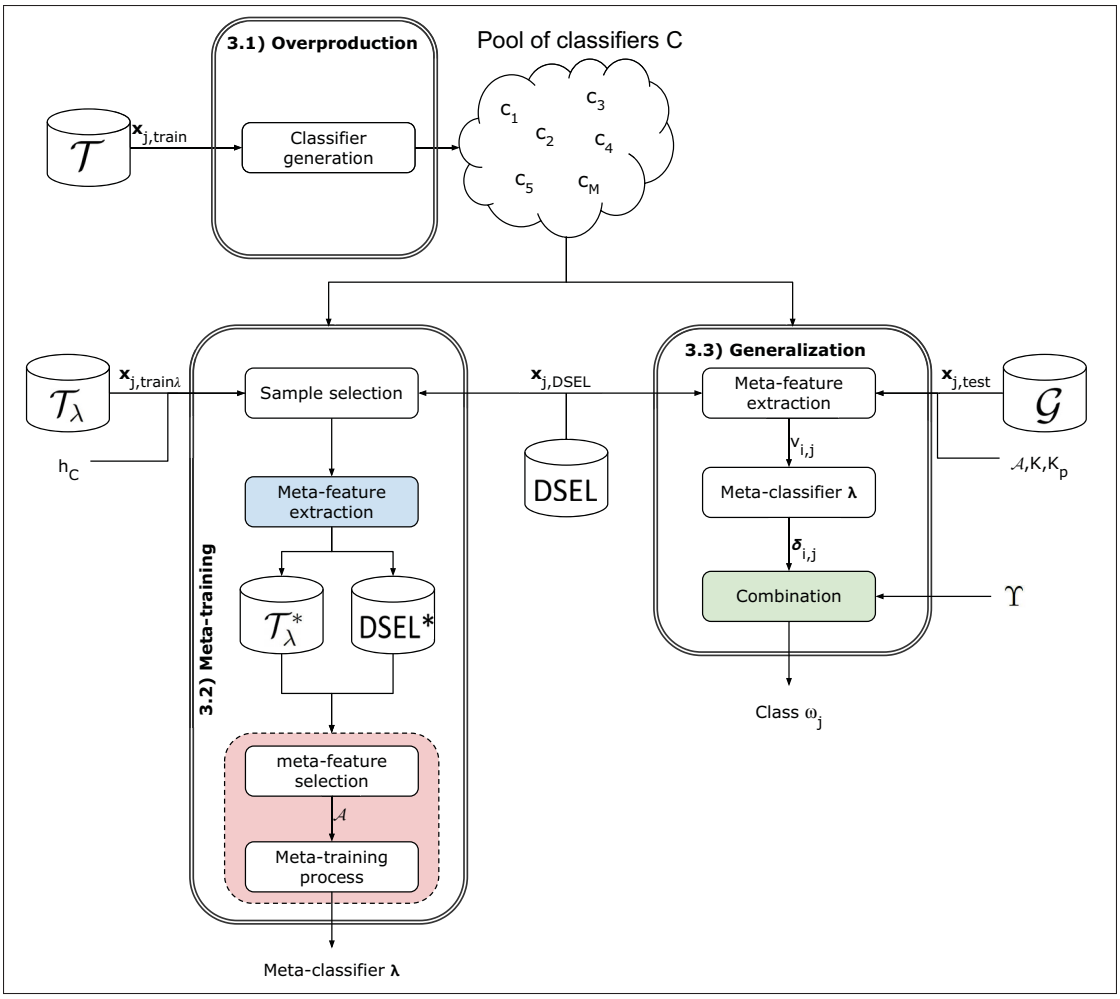


Figure 4.1 Overview of the proposed framework. It is divided into three steps: 1) Overproduction, where the pool of classifiers $C = \{c_1, \ldots, c_M\}$ is generated, 2) The training of the selector λ (meta-classifier), and 3) The generalization phase, where the level of competence $\delta_{i,j}$ of each base classifier c_i is calculated specifically for each new test sample $\mathbf{x}_{j,test}$. h_C, K, K_p and Υ are the hyper-parameters required by the proposed system

Perceptrons for the multi-class problems. The use of linear classifiers is motivated by the finding in [37; 86; 6] showing that the META-DES framework can solve complex non-linear classification problems with complex decision boundaries using only linear classifiers [37].

4.3.2 Meta-training Phase

In this stage, the meta-features are extracted for the training of the meta-classifier λ. In this version of the framework we extract meta-data from two sets: the meta-training set $\mathcal{T}_\lambda$ and the dynamic selection (validation) *DSEL*. The meta-data extracted from the set $\mathcal{T}_\lambda$, denoted by $\mathcal{T}_\lambda^*$ are used for the training of the meta-classifier. The meta-data extracted from the set *DSEL*, denoted by $DSEL^*$ are used as validation data in the BPSO optimization scheme for preventing overfitting.

4.3.2.1 Sample Selection

The first step in the meta-data generation process is the sample selection mechanism. The sample selection mechanism is employed in order to focus the training of the meta-classifier to deal with cases in which the extent of consensus of the pool is low, i.e., when there is a disagreement between the classifiers in the pool, for the correct label. For each instance $\mathbf{x}_j$[1] coming from either the meta-training set, $\mathcal{T}_\lambda$, or the dynamic selection dataset DSEL, the consensus of the pool is computed by the percentage of base classifiers in the pool that predicts its correct label, denoted by $H\left(\mathbf{x}_j, C\right)$. If the percentage falls below the consensus threshold, h_c, the sample, $\mathbf{x}_j$, is passed down to the meta-features extraction process.

Next, for each base classifier, $c_i \in C$, 15 sets of meta-features are computed. Each set of meta-features is detailed in Section 4.4. The meta-feature vector $v_{i,j}$ containing the 15 sets of meta-features is obtained at the end of the process. The meta-feature vector $v_{i,j}$ represents the behavior of the base classifier c_i for the classification of the query sample $\mathbf{x}_j$. If the base classifier c_i predicts the correct label for $\mathbf{x}_j$, the class attribute of $v_{i,j}$, $\alpha_{i,j} = 1$ ($v_{i,j}$ belongs to the meta-class "competent"), otherwise $\alpha_{i,j} = 0$ (belongs to the meta-class "incompetent"). $v_{i,j}$ is stored in either $\mathcal{T}_\lambda^*$ or DSEL*. It is important to mention that for each sample, $\mathbf{x}_j$, a total of M (M is the size of the pool of classifiers, C) meta-feature vectors $v_{i,j}$ are extracted, each one representing the behavior of a single base classifier, c_i, for the classification of the sample

[1] $\mathbf{x}_{j,DSEL}$ coming from the set DSEL or $\mathbf{x}_{j,train_\lambda}$ coming from the set $\mathcal{T}_\lambda$

$\mathbf{x}_j$. For instance, consider that 200 training samples are available for the meta-training stage ($N = 200$); if the pool C is composed of 100 classifiers ($M = 100$), the meta-training dataset is the number of training samples $N \times$ the size of the pool M, $N \times M = 20.000$. After obtaining the sets $\mathcal{T}_\lambda^*$ and DSEL*, the BPSO meta-features selection procedure is called.

4.3.2.2 Meta-Feature Selection Using Binary Particle Swarm Optimization (BPSO)

Since we are dealing with feature selection, a binary version of the PSO algorithm, BPSO, is considered. Each particle (solution) is composed of a binary string $\mathcal{S}_i = \left\{\mathcal{S}_{i,1}, ..., \mathcal{S}_{i,D}\right\}$ (D is the number of meta-features), where every bit $\mathcal{S}_{i,d}$ represents a single meta-feature. The value "1" means the meta-feature is selected and "0" otherwise.

At each generation, the velocity of the i-th particle is computed using Equation 4.1:

$$velocity_i^{g+1} = wv_i^g + c_1 \times rand \times (pBest_i - \mathcal{S}_i^g) + c_2 \times rand \times (gBest - \mathcal{S}_i^g) \qquad (4.1)$$

Each particle makes use of its private memory, $pBest_i$, which represents the best position the i-th particle visited as well as the knowledge of the swarm, $gBest$, which represent the global best position visited, considering the whole swarm. The constant w corresponds to the inertia weight, c_1 and c_2 are the acceleration coefficients, and $rand$ is a randomly generated number between 0 and 1. The term $c_1 \times rand \times (pBest_i - \mathcal{S}_i^g)$ represents the private knowledge of the i-th particle, and the term $c_2 \times rand \times (gBest - \mathcal{S}_i^g)$ represents the collaboration of particles.

When dealing with binary search spaces, updating the position of a particle means switching between "0" and "1", i.e., whether or not the meta-feature is selected. The switching is conducted based on the velocity of the particle. The higher the velocity of a particle, the higher its probability of changing positions should be. However, the velocities are computed in the real space rather than in the binary space (as shown in Equation 4.1). The velocity of the particle needs to be converted into a probability value, representing the probability of changing the position of the particle from "0" to "1" and vice versa. This step is conducted using a transfer

function, $\mathscr{T}$. A transfer function should work in a way that the higher the velocity value, the higher the probability of changing position will be, since particles with higher velocity values are probably far from the best solutions ($pBest_i$ and $gBest$). Similarly, a transfer function must present a lower probability of switching position for lower velocity values [77]. The position of the i-th particle is updated according to Equation 4.2.

$$\mathcal{S}_i^{g+1} = \begin{cases} (\mathcal{S}_i^{g+1})^{-1} & If \;\; rand < \mathscr{T}(velocity_i^d(g+1)) \\ \mathcal{S}_i^{g+1} & If \;\; rand \geq \mathscr{T}(velocity_i^d(g+1)) \end{cases} \tag{4.2}$$

Generally speaking, there are two main types of transfer functions, S-shaped and V-shaped [77]. In this work we consider one S-shaped transfer function proposed in [76] and one V-shaped transfer function proposed in [77], in Equations 4.3 and 4.4, respectively. These transfer functions were selected since they obtained the best results in several optimization benchmarks [77].

$$\mathscr{T}_S(x) = \frac{1}{1+e^{-2x}} \tag{4.3}$$

$$\mathscr{T}_V(x) = \left| \frac{2}{\pi} arc\, tan\left(\frac{\pi}{2}x\right) \right| \tag{4.4}$$

4.3.2.2.1 Fitness Function - Distance to the Oracle

The Oracle is an abstract method which always selects the classifier that predicts the correct label for the test sample $\mathbf{x}_{j,test}$ if such a classifier exists. Hence, the Oracle can be seen as an ideal classifier selection scheme. In this work, we formalize the Oracle as the ideal classifier selection technique which always selects the classifier that predicts the correct label, $\mathbf{x}_j$, and rejects otherwise (Equation 4.5).

$$\begin{cases} \delta_{i,j} = 1, & \text{if } c_i \text{ correctly classifies } \mathbf{x}_j \\ \delta_{i,j} = 0, & \text{otherwise} \end{cases} \tag{4.5}$$

In other words, the level of competence $\delta_{i,j}$ of a base classifier c_i is 1 if it predicts the correct label for $\mathbf{x}_j$, and 0 otherwise.

The fitness function is computed as follows: Given that $\delta_{i,j}^{\lambda}$ and $\delta_{i,j}^{Oracle}$ are the level of competence of the base classifier c_i for the classification of the instance, $\mathbf{x}_j$, computed by the META-DES framework and the Oracle, respectively. The distance between both techniques $d_{\lambda,Oracle}$ is calculated by the mean squared difference between $\delta_{i,j}^{\lambda}$ and $\delta_{i,j}^{Oracle}$ (Equation 4.6).

$$d_{\lambda,Oracle} = \frac{1}{NM}\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{M}\left(\delta_{i,j}^{\lambda} - \delta_{i,j}^{Oracle}\right)^2} \tag{4.6}$$

where N and M are the size of the dataset and pool of classifiers, respectively.

Therefore, the BPSO optimization searches for a meta-classifier which minimizes the distance $d_{\lambda,Oracle}$. In other words, we search for a meta-classifier λ that presents a behavior closer to the ideal dynamic selection technique, for estimating the competence level of the base classifiers. Moreover, the distance to the Oracle fitness function is motivated by the results obtained in our previous work [55], in which we demonstrated that dynamic selection techniques with smaller distances to the Oracle are more likely to achieve higher classification performance. We call the proposed system META-DES.Oracle since the formal definition of the Oracle is used for optimizing the performance of the meta-classifier.

4.3.2.2.2 Overfitting Control Scheme

Since the fitness function takes into account the performance of the meta-classifier, i.e., the wrapper approach, the optimization process becomes another learning process and may be

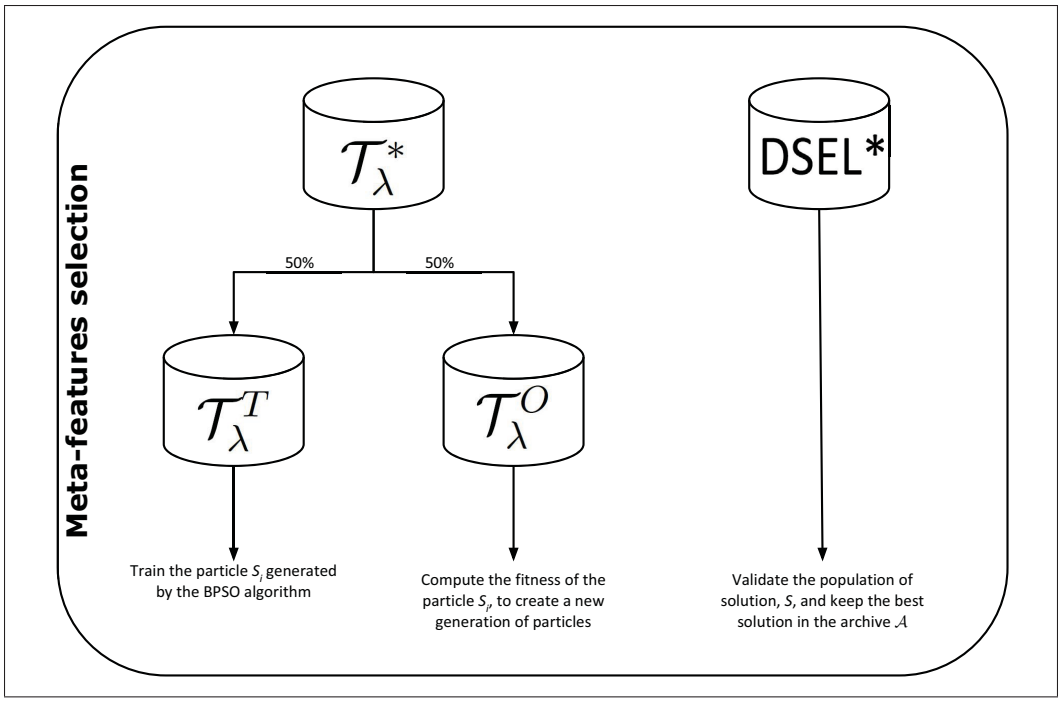


Figure 4.2 Division of the datasets for the BPSO with global validation scheme

prone to overfitting [75; 43; 87]. The best solution found during the optimization routine may have overfitted the optimization dataset, and may not have a good generalization performance. To avoid overfitting, the sets used in the BPSO feature selection scheme are divided as illustrated in Figure 4.2. The meta-feature dataset,$\mathcal{T}_\lambda^*$, is split on the basis of 50% for the training of the meta-classifier $\mathcal{T}_\lambda^T$ and 50% for the optimization dataset $\mathcal{T}_\lambda^O$ which is used to guide the search in the BPSO scheme. The meta-feature vectors extracted from the dynamic selection dataset, $DSEL^*$, are used to validate the solutions S_i found by the BPSO algorithm.

There are three common methods for controlling overfitting in optimization systems [43]: Partial Validation (PV), Backwarding Validation (BV), and Global Validation (GV). In this work, we use the GV approach since previous works in the literature demonstrate that the GV is a more robust alternative for controlling overfitting in optimization techniques [75; 43]. In the GV scheme (see Algorithm 4.1), at each generation, the fitness of all particles $S_i^g \in S$ are evaluated using the validation set, $DSEL^*$ (line 18 of the algorithm). If the fitness of the particle S_i^g is better than the fitness of the particle kept in the archive, denoted by $\mathcal{A}$, S_i^g is stored in the

archive (lines 19 and 20). Thus, at the end of the optimization process, the particle kept in $\mathscr{A}$ is the one presenting the best fitness value, considering the validation data. The solution kept in the archive, $\mathscr{A}$, is used as the meta-classifier λ.

1: $\mathscr{A} = \emptyset$
2: Randomly initialize a swarm $S = \{S_1, S_2, ..., S_{max(S)}\}$
3: **for** each generation $g \in 1, ..., max(g)$ **do**
4: "Perform all steps to generate the new solutions"
5: **for** each particle $S_i^g \mid i = 1, ..., max(S)$ **do**
6: Evaluate fitness of the particle S_i^g (Section 4.3.2.2.1).
7: **if** fitness(S_i^g) < fitness($pBest_i$) **then**
8: $pBest_i = S_i^g$
9: **end if**
10: **if** fitness(S_i^g) < fitness($gBest$) **then**
11: $gBest = S_i^g$
12: **end if**
13: **end for**
14: Compute the velocity of each particle using Equation 4.1.
15: Update the position of each particle using Equation 4.2.
16: **for** each particle $S_i^{g+1} \mid i = 1, ..., max(S)$ **do**
17: Estimate the fitness of S_i^{g+1} using the dataset $DSEL^*$.
18: **if** fitness(S_i^{g+1}) < fitness($\mathscr{A}$) **then**
19: "Store S_i^{g+1} in the archive."
20: $\mathscr{A} = S_i^{g+1}$.
21: **end if**
22: **end for**
23: **end for**
24: **return** The particle stored in the archive $\mathscr{A}$.

Algorithm 4.1: BPSO meta-features selection with Global Validation

4.3.3 Generalization Phase

Input: Query sample $\mathbf{x}_{j,test}$
Input: Pool of classifiers $C = \{c_1, \ldots, c_M\}$
Input: The solution kept in the archive $\mathscr{A}$.
Input: dynamic selection dataset *DSEL*
1: $C^{'} = \emptyset$
2: **for all** $c_i \in C$ **do**
3: Compute the meta-features selected in the archive $\mathscr{A}$ to obtain $v_{i,j}$.
4: input $v_{i,j}$ to λ
5: Estimate the level of competence $\delta_{i,j}$.
6: **if** $\delta_{i,j} \geq \Upsilon$ **then**
7: $C^{'} = C^{'} \cup \{c_i\}$
8: $\delta^{'}_{i,j} = \delta^{'}_{i,j} \cup \{\delta_{i,j}\}$
9: **end if**
10: **end for**
11: "Each selected base classifier $c_{i,j}$ is weighted by it's competence level $\delta_{i,j}$.
12: $w_l = WeightedMajorityVote(\mathbf{x}_{j,test}, C^{'}, \delta^{'}_{i,j})$
13: **return** The predicted label w_l for the sample $\mathbf{x}_{j,test}$

Algorithm 4.2: Classification steps using the selector λ

The generalization procedure is formalized by Algorithm 4.2. Given the query sample, $\mathbf{x}_{j,test}$, the region of competence θ_j is computed using the samples from the dynamic selection dataset *DSEL*. Following that, the output profiles, $\tilde{\mathbf{x}}_{j,test}$ of the test sample, $\mathbf{x}_{j,test}$, are calculated. The set with K_p similar output profiles, ϕ_j, of the query sample $\mathbf{x}_{j,test}$, is obtained through the Euclidean distance applied over the output profiles of the dynamic selection dataset.

For each base classifier, c_i, belonging to the pool of classifiers C, the meta-feature extraction process is called (Section 4.4), returning the meta-features vector $v_{i,j}$ (lines 5 and 6). Only the selected meta-features, which are kept in the archive $\mathscr{A}$ are extracted. Then, $v_{i,j}$ is used as input to the meta-classifier λ. The support, $\delta_{i,j}$, obtained by λ for the "competent" meta-class, is computed as the level of competence of the base classifier, c_i, for the classification of the test sample, $\mathbf{x}_{j,test}$. The classification of the query sample, $\mathbf{x}_{j,test}$, is performed using a hybrid dynamic selection and weighting approach. First, the base classifiers that achieve a

level of competence, $\delta_{i,j} > \Upsilon = 0.5$, are considered competent, and are selected to compose the ensemble, C' (lines 7 to 9). Next, the decision of each selected base classifier is weighted by its level of competence, $\delta_{i,j}$, using a weighted majority voting scheme (line 13) to predict the label w_l of the query sample $\mathbf{x}_{j,test}$. Thus, the base classifiers that attained a higher level of competence, $\delta_{i,j}$, have more influence in the final decision.

4.4 Meta-Feature Extraction

A total of 15 sets of meta-features are considered, with ten sets proposed in this paper, and five coming from our previous work [2]. Each set f_i captures a different property of the behavior of the base classifier, and can be seen as a different criterion to dynamically estimate the level of competence of the base classifier, such as the classification performance estimated in a local region of the feature space and the classifier confidence for the classification of the input sample. Using 15 distinct sets of meta-features, even though one criterion might fail due to imprecisions in the local regions of the feature space or due to low confidence results, the system can still achieve a good performance, as other meta-features are considered by the selection scheme.

Table 4.1 shows the criterion used by each f_i, the object used to extract the meta-feature (e.g., the region of competence, θ_j), and its categorization based on the DES taxonomy suggested in [1]. Each set of meta-features may generate more than one feature. The size of the feature vector, $v_{i,j}$, is $(K \times 8) + K_p + 6$.

Given a new sample, $\mathbf{x}_j$, the first step in extracting the meta-features involves computing its region of competence, denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. The region of competence is defined in the dynamic selection dataset *DSEL* set using the K-Nearest Neighbor algorithm. Then, $\mathbf{x}_j$ is transformed into an output profile $\tilde{\mathbf{x}}_j$. The output profile of the instance $\mathbf{x}_j$ is denoted by $\tilde{\mathbf{x}}_j = \left\{\tilde{\mathbf{x}}_{j,1}, \tilde{\mathbf{x}}_{j,2}, \ldots, \tilde{\mathbf{x}}_{j,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,i}$ is the decision yielded by the base classifier c_i for the sample $\mathbf{x}_j$ [16]. Then, the similarity between $\tilde{\mathbf{x}}_j$ and the output profiles of the samples in *DSEL* is obtained through the Euclidean distance. The K_p most similar output profiles are

Table 4.1 A summary of each set of meta-features. They are categorized into the subgroups proposed in [1]. K is the size of the region of competence, θ_j, and K_p the size of the output profiles set ϕ_j containing the K_p most similar output profiles of the query sample $\mathbf{x}_j$. The size of the meta-feature vector is $(K \times 8) + K_p + 6$. The sets of meta-features marked with an * correspond to sets previously defined in [2]

Meta-Feature	Criterion	Domain	Object	No. of Features
f_{Hard}*	Classification of the K-Nearest Neighbors	Accuracy	θ_j	K
f_{Prob}*	Posterior probability obtained for the K-Nearest Neighbors	Probabilistic	θ_j	K
$f_{Overall}$*	Overall accuracy in the region of competence	Accuracy	θ_j	1
f_{Cond}	Conditional accuracy in the region of competence	Accuracy	θ_j	1
f_{Conf}*	Degree of confidence for the input sample	Confidence	$\mathbf{x}_j$	1
f_{Amb}	Ambiguity in the vector of class supports	Ambiguity	$\mathbf{x}_j$	1
f_{Log}	Logarithmic difference between the class supports	Probabilistic	$S(\mathbf{x}_j)$	K
f_{PRC}	Probability of Random Classifier	Probabilistic	$S(\mathbf{x}_j)$	K
f_{MD}	Minimum difference between the predictions	Probabilistic	$S(\mathbf{x}_j)$	K
f_{Ent}	Entropy in the vector of class supports	Probabilistic	$S(\mathbf{x}_j)$	K
f_{Exp}	Exponential difference between the class supports	Probabilistic	$S(\mathbf{x}_j)$	K
f_{KL}	Kullback-Leibler divergence	Probabilistic	$S(\mathbf{x}_j)$	K
f_{OP}*	Output profiles classification	Behavior	ϕ_j	K_p
f_{Rank}	Classifier ranking in the feature space	Ranking	DSEL	1
$f_{Rank_{OP}}$	Classifier ranking in the decision space	Behavior and Ranking	ϕ_j	1

selected to form the set $\phi_j = \left\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{K_p}\right\}$, where each output profile $\tilde{\mathbf{x}}_k$ is associated with a label $w_{l,k}$.

4.4.1 Local Accuracy Meta-Features

These meta-features are based on the performance of the base classifier in a local region of the feature space surrounding the query instance $\mathbf{x}_j$. Three sets of meta-features using local accuracy estimation are considered:

4.4.1.1 Overall Local accuracy: $f_{Overall}$

The accuracy of c_i over the whole region of competence θ_j is computed and encoded as $f_{Overall}$ (Equation 4.7).

$$f_{Overall} = \sum_{k=1}^{K} P(w_l \mid \mathbf{x}_k \in w_l, c_i) \tag{4.7}$$

4.4.1.2 Conditional Local Accuracy: f_{cond}

The local accuracy of c_i is estimated with respect to the output classes; w_l (w_l is the class assigned for $\mathbf{x}_j$ by c_i) for the samples belonging to the region of competence, θ_j (Equation 4.8).

$$f_{cond} = \frac{\sum_{\mathbf{x}_k \in w_l} P(w_l \mid \mathbf{x}_k, c_i)}{\sum_{k=1}^{K} P(w_l \mid \mathbf{x}_k, c_i)} \tag{4.8}$$

4.4.1.3 Neighbors' hard classificationL: f_{Hard}

First, a vector with K elements is created. For each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of the vector is set to 1, otherwise it is 0. Thus, K meta-features are computed.

4.4.2 Ambiguity

Ambiguity measures the level of confidence the base classifier c_i has in its answer. A common concept used to estimate the confidence of a classifier is based on the margin theory [5; 88]. The margin of a classifier is regarded as a good indicator of the classifier's confidence. Two meta-features are considered: one based on the maximum margin theory f_{conf} and one based on the minimum margin theory f_{amb}. Since these meta-features do not take into account the correct label of the sample, they are extracted directly from the query sample, $\mathbf{x}_j$.

4.4.2.1 Classifier's confidence: f_{Conf}

The perpendicular distance between the input sample, $\mathbf{x}_j$, and the decision boundary of the base classifier c_i is calculated and encoded as f_{conf}. The value of f_{conf} is normalized to a $[0-1]$ range using the Min-max normalization.

4.4.2.2 Ambiguity: f_{Amb}

This information is simply computed by the difference between scores of the class with highest support and the second highest one for the query sample, $\mathbf{x}_j$, e.g., consider that for a 3-class classification problem, the scores obtained by the base classifier c_i for a given query sample, $\mathbf{x}_j$, are 0.65, 0.30 and 0.05. Then, the ambiguity value is $f_{amb} = 0.65 - 0.30 = 0.35$. A higher value in f_{amb} means that the classifier decision is less ambiguous.

4.4.3 Probabilistic Meta-Features

This class of meta-features is based on probabilistic models that are applied over the vector of class supports produced by the base classifier c_i for the classification of a given query sample. The motivation behind probabilistic measures derives from the observation that classifiers that perform worse than the random classifier, i.e., a classifier that randomly select the classes with equal probabilities, deteriorate the majority voting performance. In contrast, if the base classifiers are significantly better than the random classifier, they are likely to improve the majority voting accuracy [45]. Hence, each set of meta-features in this group estimates the probability that the performance of a given base classifier c_i is significantly different from that of a random classifier derived from different probabilistic and information theory perspectives [18; 40; 45; 41; 46].

For the definitions below, let $S(\mathbf{x}_k) = \{S_1(\mathbf{x}_k), \ldots, S_L(\mathbf{x}_k)\}$ be the vector of class supports estimated by the base classifier c_i for a given sample, $\mathbf{x}_k$, where each value $S_l(\mathbf{x}_k), l = 1, 2 \ \ldots, \ L$ represents the support given to the l-th class and $\sum_{l=1}^{L} S_l(\mathbf{x}_k) = 1$. Let $S_{lk}(\mathbf{x}_j)$ be the support given by the base classifier c_i for the correct class label of $\mathbf{x}_j$. The output of the random classifier follows a uniform distribution, and is denoted by $RC = \left\{\frac{1}{L}, ..., \frac{1}{L}\right\}$.

4.4.3.1 Posterior probability: f_{Prob}

First, a vector with K elements is created. Then, for each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, the posterior probability of c_i, $P(w_l \mid \mathbf{x}_k)$ is computed and inserted into the k-th position of the vector. Consequently, K meta-features are computed.

4.4.3.2 Logarithmic: f_{Log}

First, a vector with K elements is created, $f_{Log} = \left\{f_{Log}(1),\ ...,\ f_{Log}(K)\right\}$. For each instance, $\mathbf{x}_k$, belonging to the region of competence θ_j, the support obtained by the base classifier c_i for the correct class label, $S_{lk}(\mathbf{x}_k)$, is estimated. Then, a logarithmic function is applied to $S_{lk}(\mathbf{x}_k)$ (Equation 4.9). The logarithmic function is used such that the value of the meta-feature is negative if the support obtained for the correct class label is lower than the support obtained from random guessing (i.e., $S_{lk}(\mathbf{x}_j) < \frac{1}{L}$) and positive otherwise. The result of the logarithmic function is inserted into the k-th position of the vector. Hence, K meta-features are computed.

$$f_{log}(k) = 2 \times S_{lk}(\mathbf{x}_k)^{\frac{log(2)}{log(L)}} - 1 \tag{4.9}$$

4.4.3.3 Entropy: f_{Ent}

The entropy measures the degree of uncertainty in the vector of supports, $S(\mathbf{x}_k)$, obtained by the base classifier, c_i. The meta-feature is calculated as follows: first, a vector with K elements is created, $f_{Ent} = \{f_{Ent}(1),\ ...,\ f_{Ent}(K)\}$. Then, for each instance, $\mathbf{x}_k$, belonging to the region of competence, θ_j, the entropy of the vector of class supports is computed, and inserted in the k-th position of the vector f_{Ent} (Equation 4.10). Thus, K meta-features are computed.

$$f_{Ent}(k) = -\sum_{l=1}^{L} S_l(\mathbf{x}_k) log(S_l(\mathbf{x}_k)) \tag{4.10}$$

4.4.3.4 Minimal difference: f_{MD}

First, a vector with K elements is created, $f_{MD} = \{f_{MD}(1),\ ...,\ f_{MD}(K)\}$. Then, for each sample, $\mathbf{x}_k$, belonging to the region of competence, θ_j, the difference between the support obtained by the base classifier c_i for the correct class label of $\mathbf{x}_k$, $S_{lk}(\mathbf{x}_k)$, and those obtained by c_i for each of the other classes, $S_l(\mathbf{x}_k) \mid\ l \neq lk$, are calculated. The difference which produces the minimal value is inserted in the k-th position of the vector f_{MD} (Equation 4.11). Thus, K meta-features are computed.

$$f_{MD}(k) = min_{l \in L,\, l \neq lk}\,[S_l(\mathbf{x}_k) - S_{lk}(\mathbf{x}_k)] \tag{4.11}$$

4.4.3.5 Kullback-Leibler Divergence: f_{KL}

The Kullback-Leibler (KL) divergence [89] estimates the competence of a base classifier c_i from the information theory perspective [45]. The meta-feature is computed as follows: first, a vector with K elements is created, $f_{KL} = \{f_{KL}(1),\ ...,\ f_{KL}(K)\}$. Then, for each member, $\mathbf{x}_k$, belonging to the region of competence θ_j, the KL divergence between the vector of class supports, $S(\mathbf{x}_k) = \{S_1(\mathbf{x}_k), \ldots, S_L(\mathbf{x}_k)\}$, obtained by the base classifier, c_i, and those obtained by the random classifier, $RC = \left\{\frac{1}{L}, ..., \frac{1}{L}\right\}$ is computed. The result of the KL divergence is inserted in the k-th position of the vector f_{KL} (Equation 4.12). Consequently, K meta-features are calculated.

$$f_{KL}(k) = \sum_{l=1}^{L} S_l(\mathbf{x}_k) log \frac{S_l(\mathbf{x}_k)}{RC} \tag{4.12}$$

4.4.3.6 Exponential: f_{Exp}

First, a vector with K elements is created, $f_{Exp} = \left\{f_{Exp}(1),\ ...,\ f_{Exp}(K)\right\}$. For each sample, $\mathbf{x}_k$, belonging to the region of competence θ_j, the support obtained by the base classifier c_i

for the correct class label, $S_{lk}(\mathbf{x}_k)$, is estimated. Next, an exponential function is applied over $S_{lk}(\mathbf{x}_k)$ to compute f_{Exp} (Equation 4.13). Using the exponential function, the value of f_{Exp} increases exponentially when the value of $S_{lk}(\mathbf{x}_k)$ is higher than that obtained from random guessing ($S_{lk}(\mathbf{x}_k) > \frac{1}{L}$), and is negative otherwise. The result of the exponential function is inserted in the k-th position of the vector. Hence, K meta-features are computed.

$$f_{Exp}(k) = 1 - 2^{-1\frac{(L-1)S_{lk}(x_k)}{1-S_{lk}(x_k)}} \tag{4.13}$$

4.4.3.7 Randomized Reference Classifier: f_{PRC}

First, a vector with K elements is created, $f_{PRC} = \{f_{PRC}(1),\ ...,\ f_{PRC}(K)\}$. For each sample, $\mathbf{x}_k$, belonging to the region of competence θ_j, the conditional probability of correct classification estimated by the randomized reference classifier (RRC) proposed in [18] is estimated [2]. The result is inserted in the k-th position of the vector. Thus, K meta-features are computed.

4.4.4 Behavior meta-features

These measures take into consideration information extracted from the decision space, i.e., the outputs or behavior of the classifiers in the pool, rather than information from the feature space. Global information about the whole pool of classifiers is considered. Furthermore, many authors have successfully utilized DES criteria based on classifier behavior in estimating the competence of base classifiers [16; 17; 2].

4.4.4.1 Output profiles classification: f_{OP}

First, a vector with K_p elements is created. Then, for each member, $\tilde{\mathbf{x}}_k$, belonging to the set of output profiles, ϕ_j, if the label produced by c_i for $\mathbf{x}_k$ is equal to the label $w_{l,k}$ of $\tilde{\mathbf{x}}_k$, the k-th position of the vector is set to 1, otherwise it is 0. A total of K_p meta-features are extracted.

[2] Matlab code for this technique is available on: http://www.mathworks.com/matlabcentral/fileexchange/28391-a-probabilistic-model-of-classifier-competence

4.4.5 Ranking Meta-Features

Ranking methods for estimating the competence of base classifiers have been proposed in [38]. The ranking is computed such that classifiers with higher ranking values are more likely to be competent. In this work, we consider two types of ranking meta-features, one based on the feature space, and the other on the decision space. They are defined below:

4.4.5.1 Simplified classifier rank: f_{Rank}

This meta-feature is inspired by the simplified classifier rank technique proposed in [22]. The first step in extracting the ranking meta-feature is to order the instances in *DSEL* by its distance to the query sample $\mathbf{x}_j$. f_{Rank} is computed as the number of consecutive correct predictions made by the base classifier c_i, starting from the closest sample to $\mathbf{x}_j$. The search stops when the first misclassification is made.

4.4.5.2 classifier rank OP: $f_{Rank_{OP}}$

This meta-feature is computed similarly to the previous f_{rank}. However the search is conducted in the decision space, using the output profiles, ϕ_j, rather than the dataset *DSEL*. Hence, the first step is to order the output profiles in ϕ_j by their similarity to the output profile of the query sample $\tilde{\mathbf{x}}_j$. Then, the number of consecutive correct predictions made by the base classifier c_i is computed as $f_{Rank_{OP}}$.

4.5 Case study using synthetic data

In this section, we conduct experiments using a synthetic dataset in order to illustrate the benefits of the meta-feature selection process and compare different versions of the META-DES framework for solving a problem with a complex non-linear geometry using a pool composed of linear classifiers. The P2 is a two-class problem, presented by Valentini [67], in which each class is defined in multiple decision regions delimited by polynomial and trigonometric functions (Equations 4.14, 4.15, 4.16 and 4.17). As in [68], $E4$ was modified such that the area of

each class is approximately equal. The P2 problem is illustrated in Figure 4.3. It is impossible to solve this problem using a single linear classifier, and the performance of the best possible linear classifier is around 50%.

$$E1(x) = sin(x) + 5 \tag{4.14}$$

$$E2(x) = (x-2)^2 + 1 \tag{4.15}$$

$$E3(x) = -0.1 \cdot x^2 + 0.6sin(4x) + 8 \tag{4.16}$$

$$E4(x) = \frac{(x-10)^2}{2} + 7.902 \tag{4.17}$$

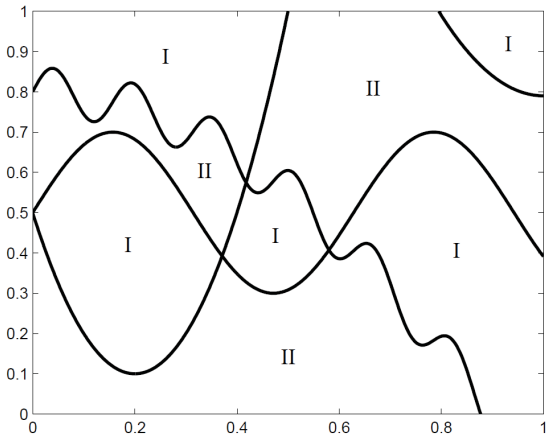


Figure 4.3 The P2 Problem. The symbols I and II represent the area of the classes, 1 and 2, respectively

For this illustrative example, the P2 problem was generated as in [37]: 500 samples for training ($\mathcal{T}$), 500 instances for the meta-training dataset ($\mathcal{T}_\lambda$), 500 instances for the dynamic selection dataset *DSEL*, and 2000 samples for the test set, $\mathcal{G}$. The pool of classifiers is composed of 5 Perceptrons (shown in Figure 4.4). The best classifier of the pool (Single Best) achieves an accuracy of 53.5%. The performance of all other base classifiers is around the 50% mark. The Oracle result of this pool obtained a recognition performance of 99.5%. In other words, there

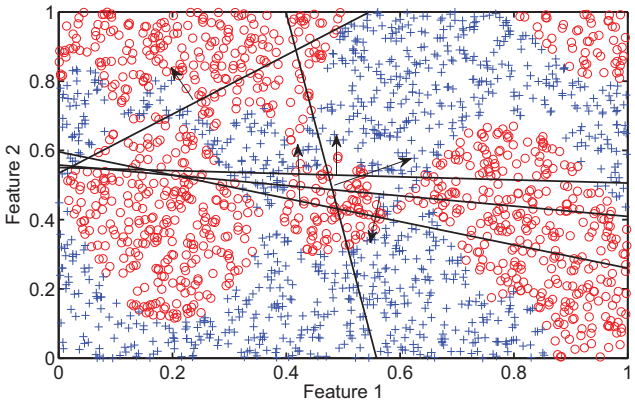
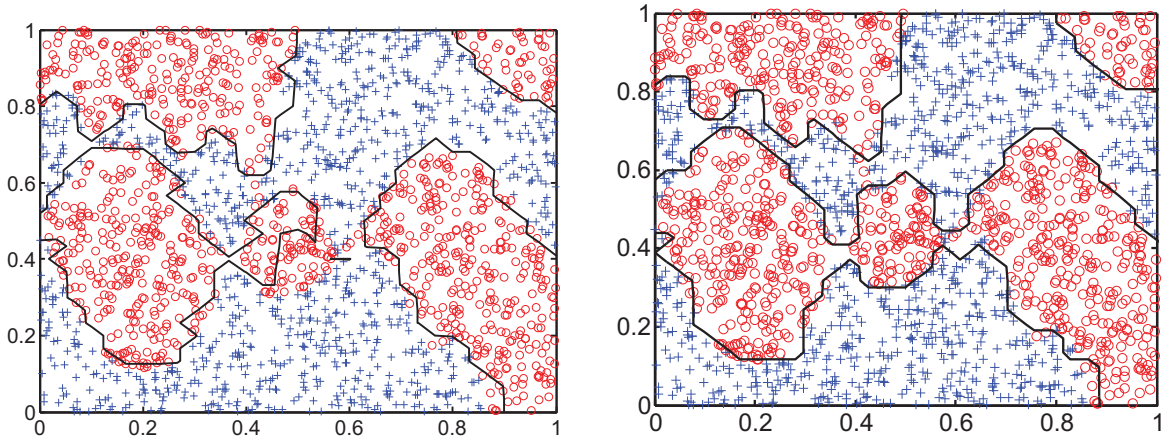


Figure 4.4 Five Perceptrons generated using the bagging technique for the P2 Problem. The arrows in each Perceptron point to the region of class 1 (red circle)

is at least one base classifier that predicts the correct label for 99.5% of est instances. The problem lies in selecting the competent classifiers in order to achieve a classification accuracy close to the Oracle.



a) META-DES b) META-DES.Oracle

Figure 4.5 Decision boundary obtained by two versions of the META-DES framework. (a) META-DESs (b) META-DES.Oracle

Figures 4.5 (a) and (b) show the decision boundary obtained by the META-DES [2], and the proposed META-DES.Oracle, respectively[3]. We can observe that the META-DES.Oracle obtains a really good approximation of the real decision boundary for the P2 problem. The META-DES.Oracle proposed in this paper obtained a recognition accuracy of 97%, while the accuracy of the META-DES was 94.5% [37]. Using the extended sets of meta-features and the meta-feature selection procedure based on the Oracle definition, we observed a significant gain in performance for the P2 problem. Thus, it is possible to reduce the big gap that exists between the performances of the current state-of-the-art DES techniques and the ideal one, the Oracle.

4.6 Experiments

4.6.1 Datasets

The experiments were conducted on the same 30 classification datasets used in our previous work [2]. Sixteen are taken from the UCI machine learning repository [59], four from the STATLOG project [60], four from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [61], four from the Ludmila Kuncheva Collection of real medical data [62], and two artificial datasets generated with the Matlab PRTOOLS toolbox [63]. The key features of each dataset are shown in Table 4.2.

4.6.2 Experimental protocol

For the sake of simplicity, the experimental setup from our previous work [2] was used. The experiments were carried out using 20 replications. For each replication, the datasets were randomly divided on the basis of 50% for training, 25% for the dynamic selection dataset, $DSEL$, and 25% for the test set, $\mathcal{G}$. The divisions were performed while maintaining the prior

[3]The results achieved by different dynamic and static ensemble techniques for the P2 problem are presented in the following report [37].

Table 4.2 Key features of the datasets used in the experiments

Database	No. of Instances	Dimensionality	No. of Classes	Source
Pima	768	8	2	UCI
Liver Disorders	345	6	2	UCI
Breast (WDBC)	568	30	2	UCI
Blood transfusion	748	4	2	UCI
Banana	1000	2	2	PRTOOLS
Vehicle	846	18	4	STATLOG
Lithuanian	1000	2	2	PRTOOLS
Sonar	208	60	2	UCI
Ionosphere	315	34	2	UCI
Wine	178	13	3	UCI
Haberman's Survival	306	3	2	UCI
Cardiotocography (CTG)	2126	21	3	UCI
Vertebral Column	310	6	2	UCI
Steel Plate Faults	1941	27	7	UCI
WDG V1	5000	21	3	UCI
Ecoli	336	7	8	UCI
Glass	214	9	6	UCI
ILPD	583	10	2	UCI
Adult	48842	14	2	UCI
Weaning	302	17	2	LKC
Laryngeal1	213	16	2	LKC
Laryngeal3	353	16	3	LKC
Thyroid	215	5	3	LKC
German credit	1000	20	2	STATLOG
Heart	270	13	2	STATLOG
Satimage	6435	19	7	STATLOG
Phoneme	5404	6	2	ELENA
Monk2	4322	6	2	KEEL
Mammographic	961	5	2	KEEL
MAGIC Gamma Telescope	19020	10	2	KEEL

probabilities of each class. For the proposed META-DES-Oracle, 25% of the training data was used in the meta-training process $\mathcal{T}_\lambda$.

For the two-class classification problems, the pool of classifiers was composed of 100 Perceptrons generated using the Bagging technique. For the multi-class problems, the pool of classifiers was composed of 100 multi-class Perceptrons. The use of linear Perceptron classifiers was motivated by the results reported in Section 4.5 showing that the META-DES framework can solve non-linear classification problems with complex decision boundaries using only a few linear classifiers. The values of the hyper-parameters, K, K_p and h_c, were set at 7, 5 and 70%, respectively. They were selected empirically based on previous publications [20; 36; 2]. Hence, the size of the meta-feature vector is 67 $((7 \times 8) + 5 + 6)$.

The parameters of the BPSO algorithm were set based on previous work in the literature [78; 83; 84]: the population size was set at 20, the maximum number of generations $max(g) = 100$. The weight function, $w = 1.0$, and acceleration coefficients, $c_1 = c_2 = 2.0$, were set using

the standard values from [76]. Moreover, the optimization process was stopped if the fitness of the best solution *gBest* failed to improve after 5 consecutive iterations. Since the BPSO optimization process is a stochastic algorithm, for each replication, the BPSO was run 30 times. The best result, considering the Global Validation overfitting control scheme, was used for generalization phase.

4.6.3 Analysis of the selected meta-features

In this section, we analyze the set of meta-features that are selected by the proposed technique. The objective of this analysis is: (1) to verify whether different sets of meta-features are better suited for different classification problems; and (2) to identify whether or not the proposed sets of meta-features are relevant.

In the first analysis, we compare how often each individual meta-feature was selected. Figure 4.6 illustrates the selection frequency per meta-feature, considering 20 replications. We present the results for each dataset separately. Each square represents an individual meta-feature. The color of each square represents the frequency that each meta-feature is selected. A white square indicates that the corresponding meta-feature was selected less than 25% of the time. A light grey square means the meta-feature was selected with a frequency between 25% and 50%. A dark grey square represents a frequency of 50% to 75%, and a black square represents a frequency of selection higher than 75%.

It can be seen that the frequency at which each meta-feature is selected varies considerably between different datasets. For instance, the meta-feature based on the classification of the neighbor samples, f_{hard}, was selected with a frequency between 25 and 50% in the majority of datasets. However, for the Wine dataset, it was not selected at all. The only exceptions are for the meta-feature sets, f_{OP}, which presented a 100% frequency of selection for all 30 datasets, and f_{cond}. This finding demonstrates that distinct classification problems require a different set of meta-features in order to better address the behavior of the Oracle. Different problems are associated with different degrees of data complexity [34], and may require a

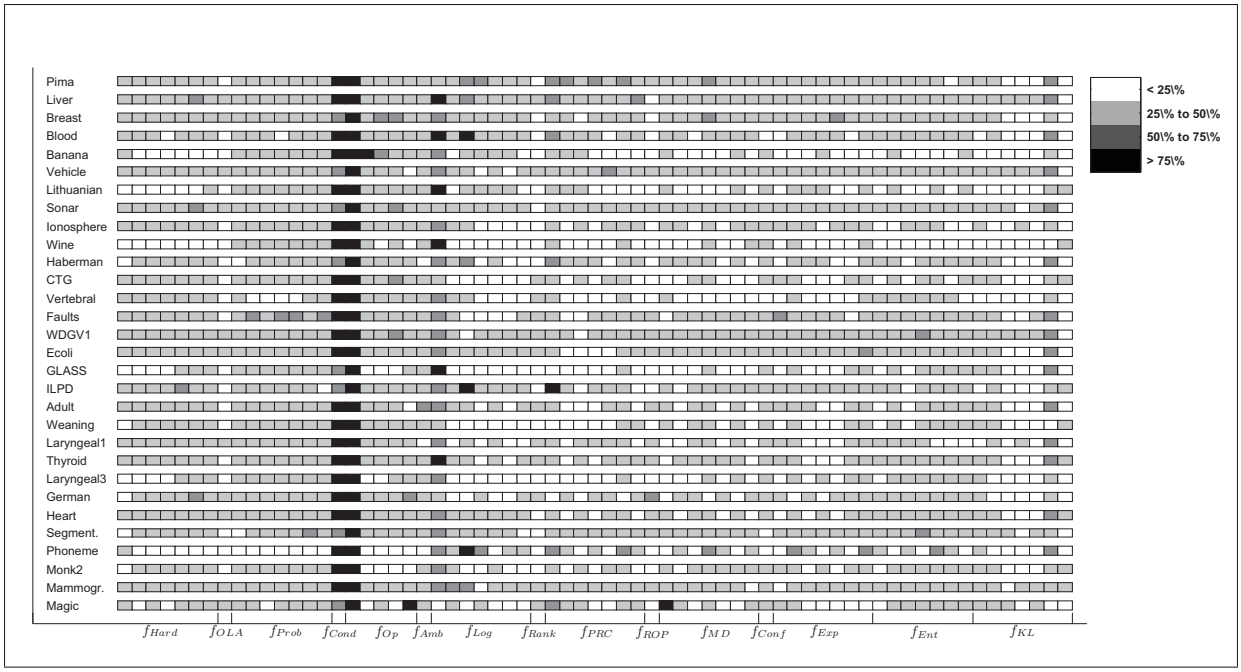


Figure 4.6 The frequency at which each individual meta-feature is selected over 20 replications. Each dataset is evaluated separately. The color of each square represents the frequency at which each meta-feature is selected. A white square indicates that the corresponding meta-feature was selected less than 25% of the time. A light grey square means the meta-feature was selected with a frequency between 25% and 50%. A dark grey square represents a frequency of 50% to 75%, and a black square represents a frequency of selection higher than 75%

distinct set of meta-features in order to obtain a meta-classifier that presents a behavior closer to the Oracle for estimating the competence of the base classifiers. Hence, the results show that the choice of the best set of meta-features is problem-dependent. In addition, we can see that each individual meta-feature is selected for at least 20% of the datasets, considering all 30 classification problems (Figure 4.7). Hence, we believe that all sets of meta-features proposed in this work are relevant.

4.6.4 Comparative study

In this section, we compare the results obtained by the proposed META-DES.Oracle, which is based on 15 sets of meta-features, against the previous versions of the META-DES framework, which are based only on five sets of meta-features defined in [2]. The objective of this com-

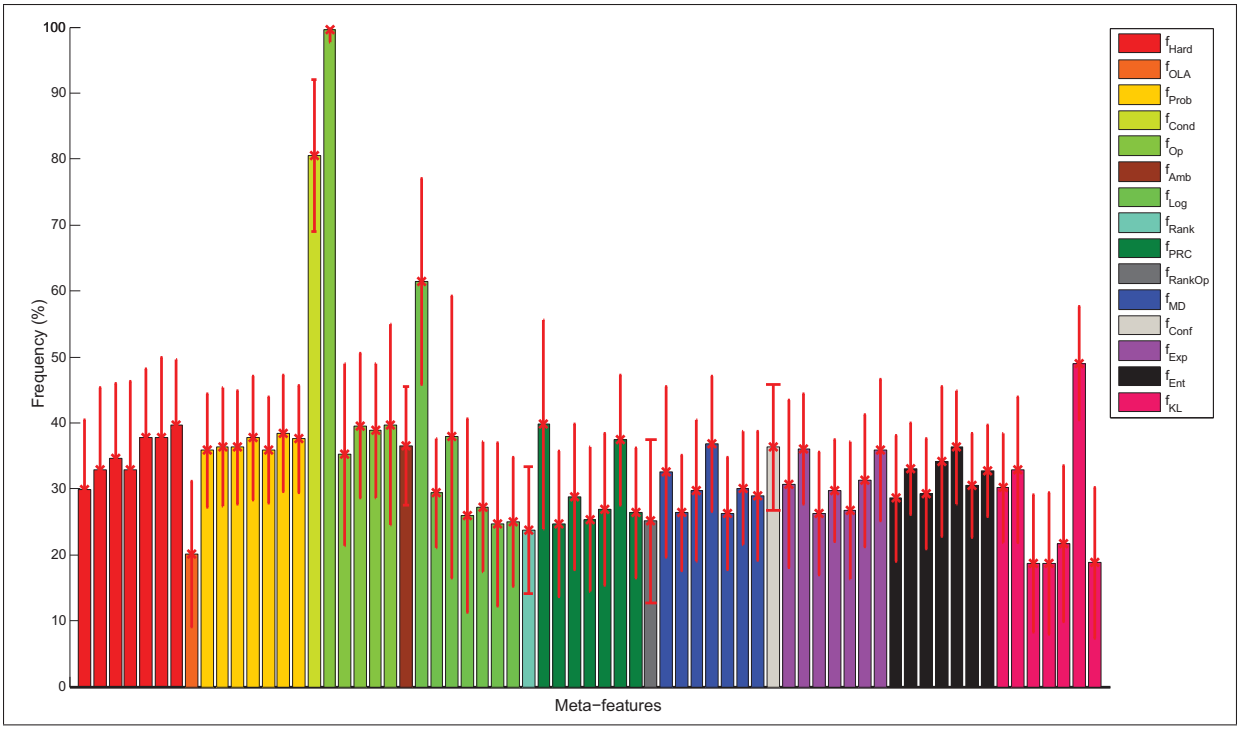


Figure 4.7 Average frequency and standard deviation per meta-feature considering 30 classification problems

parative study is to answer the following research questions: (1) Does the optimization based on the Oracle behavior lead to a significant gain in classification accuracy? (2) Does the use of more meta-features lead to a more robust DES system?

The following versions of the META-DES framework are compared in this section:

a. **S-shaped GV:** The proposed META-DES.Oracle using S-shaped transfer function with global validation.

b. **V-shaped GV:** The proposed META-DES.Oracle using V-shaped transfer function with global validation.

c. **S-Shaped:** The proposed META-DES.Oracle using S-shaped transfer function without global validation.

d. **V-Shaped:** The proposed META-DES.Oracle using V-shaped transfer function without global validation.

e. **META-DES.ALL:** The framework using the 15 sets of meta-features proposed in this work without the optimization process.

f. **META-DES.H:** The Hybrid version, META-DES.H proposed in [69].

g. **META-DES:** The first version of the META-DES framework [2].

Table 4.3 Comparison of different versions of the META-DES framework. We present the results of statistical tests at the end of the table

Dataset	S-Shaped GV	V-Shaped GV	S-Shaped	V-Shaped	META-DES.ALL	META-DES.H	META-DES
Pima	77.35(2.43)	77.53(2.24)	77.06(2.86)	77.00(2.79)	**78.34(3.26)**	77.93(1.86)	77.76(1.75)
Liver	71.50(4.96)	**72.02(4.72)**	71.24(4.94)	71.11(5.70)	68.79(4.76)	69.69(4.68)	69.56(4.84)
Breast	96.78(0.82)	96.71(0.86)	96.71(0.86)	96.71(0.86)	96.86(0.85)	97.25(0.47)	**97.41(0.50)**
Blood	79.44(1.84)	79.38(1.76)	79.79(1.38)	79.20(1.69)	**79.91(0.79)**	78.25(1.37)	78.31(1.52)
Banana	94.66(1.09)	94.54(1.16)	94.39(1.14)	94.80(0.99)	**95.69(1.35)**	94.51(2.36)	94.42(2.37)
Vehicle	82.76(1.10)	82.87(1.64)	82.61(1.48)	82.82(1.23)	81.82(1.94)	83.55(2.10)	**83.55(2.01)**
Lithuanian	95.12(2.10)	94.97(2.00)	95.49(2.21)	95.04(2.34)	**95.78(2.13)**	93.26(3.22)	93.12(3.09)
Sonar	80.13(3.96)	81.63(3.90)	81.84(4.59)	81.42(4.30)	**82.91(4.59)**	82.06(5.09)	81.84(5.67)
Ionosphere	89.31(2.26)	**89.94(1.97)**	88.80(2.60)	89.56(2.20)	89.94(2.48)	89.06(2.21)	89.06(2.21)
Wine	99.02(1.61)	**99.52(1.11)**	99.27(1.61)	99.27(1.17)	99.52(1.11)	98.53(1.48)	98.53(1.48)
Haberman	73.35(3.32)	72.03(2.67)	73.06(2.97)	72.76(3.29)	74.22(2.85)	**76.13(2.06)**	76.13(2.06)
CTG	86.73(1.23)	86.37(1.10)	86.81(1.06)	86.68(1.16)	**87.10(0.99)**	86.08(1.24)	86.04(1.14)
Vertebral	85.47(3.21)	84.90(5.33)	85.05(4.71)	84.90(6.15)	**86.47(2.38)**	84.90(2.95)	85.62(2.35)
Faults	**69.52(0.95)**	69.32(1.18)	68.93(1.15)	69.02(1.46)	69.02(1.55)	68.95(1.04)	68.72(1.19)
WDVG1	84.70(0.39)	84.72(0.49)	84.75(0.52)	84.75(0.45)	83.30(0.82)	84.77(0.65)	**84.84(0.60)**
Ecoli	**81.83(3.00)**	81.57(3.47)	81.83(3.22)	81.44(3.63)	78.70(3.22)	80.66(3.48)	80.92(3.76)
GLASS	67.09(3.89)	66.46(4.22)	66.88(3.71)	66.04(4.12)	**68.77(3.71)**	65.21(3.53)	65.21(3.65)
ILPD	68.42(2.20)	69.79(3.15)	68.04(2.74)	68.65(3.13)	69.79(3.29)	69.64(2.47)	**70.17(2.33)**
Adult	87.29(2.02)	**87.74(2.04)**	87.67(2.13)	87.67(2.03)	85.17(3.15)	87.29(1.80)	87.22(1.84)
Weaning	81.29(3.43)	**81.73(3.14)**	80.86(3.75)	81.44(3.23)	80.71(3.89)	79.98(3.55)	79.69(3.71)
Laryngeal1	86.16(4.00)	**87.42(2.98)**	85.95(3.59)	86.58(3.24)	85.11(4.33)	87.21(5.35)	87.00(5.00)
Thyroid	96.60(1.12)	96.99(0.75)	96.60(0.77)	96.86(0.91)	96.60(0.77)	**97.38(0.67)**	97.38(0.67)
Laryngeal3	**74.67(1.66)**	73.67(2.14)	74.17(2.25)	73.79(2.03)	71.67(3.34)	73.54(1.66)	73.42(1.26)
German	75.03(1.99)	**76.58(1.99)**	75.43(1.92)	76.05(1.67)	71.56(2.91)	74.36(1.28)	74.54(1.30)
Heart	85.13(2.94)	**86.44(3.38)**	85.62(3.03)	85.13(2.75)	84.15(4.35)	85.46(2.70)	85.30(2.30)
Satimage	96.59(0.68)	**96.65(0.83)**	96.50(0.82)	96.55(0.80)	96.46(0.78)	96.46(0.79)	96.42(0.76)
Phoneme	84.76(0.77)	85.05(1.08)	84.62(0.95)	85.16(1.16)	**85.22(0.88)**	81.82(0.69)	81.77(0.72)
Monk2	94.15(2.18)	**94.45(1.88)**	94.35(1.72)	94.45(1.88)	92.91(1.84)	83.45(3.46)	83.34(3.32)
Mammographic	80.35(2.85)	80.72(2.56)	81.31(3.42)	79.92(3.44)	81.15(1.58)	84.30(2.27)	**84.41(2.54)**
Magic	85.69 (1.37)	**86.02 (2.20)**	85.79 (1.21)	85.80(2.54)	85.25(3.21)	85.650(2.27)	84.35(3.27)
Average rank	3.80(0.78)	**3.00(0.92)**	4.03(0.90)	4.16(0.82)	3.96(1.26)	4.33(0.93)	4.70(1.17)
Win-Tie-Loss	17-3-10	19-9-2	16-4-10	17-2-11	15-1-14	n/a	n/a
Wilcoxon Signed Test	$\sim$ ($\rho = .3044$)	$+$ ($\rho = .0316$)	$\sim$ ($\rho = .3389$)	$\sim$ ($\rho = .2623$)	$\sim$ ($\rho = .8612$)	n/a	n/a

Classification accuracies are reported in Table 4.3. The best result achieved for each dataset is highlighted in bold. The Friedman [90] test is used in order to compare the results of all techniques over the 30 classification datasets. The Friedman test is a non-parametric equivalent

of the repeated ANOVA measures, used to make comparison between several techniques over multiple datasets [91]. For each dataset, the Friedman test ranks each algorithm, with the best performing one getting rank 1, the second best rank 2, and so forth. Then, the average rank and its standard deviation are computed, considering all datasets. The best algorithm is the one presenting the lowest average rank. Since we are comparing seven techniques, the degree of freedom is 6. We set the level of significance $\alpha = 0.05$, i.e., 95% confidence. The Friedman test shows that there is a significant difference between the seven approaches. Then, a post-hoc Bonferroni-Dunn test was conducted for a pairwise comparison between the ranks achieved by each technique. The performance of two classifiers is significantly different if their difference in average rank is higher than the critical difference. The critical difference is computed using the following equation: $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$, where the critical value q_α is based on the Studentized range statistic divided by $\sqrt{2}$. The results of the post-hoc test are presented using the critical difference diagram proposed in [91] (Figure 4.8). The performance of techniques in which the difference in average ranks is higher than the critical difference are considered significantly different. Techniques with no statistical difference are connected by a black bar in the CD diagram.

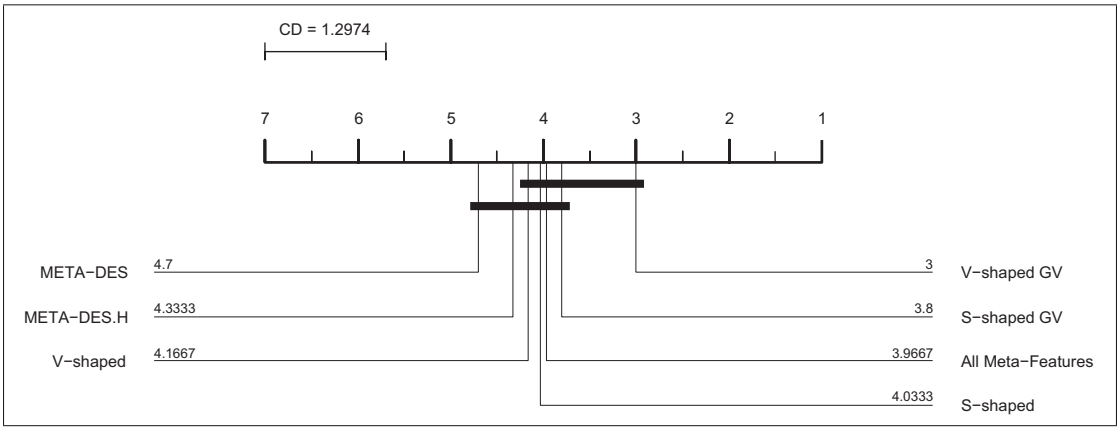


Figure 4.8 Graphical representation of the average rank for each DES technique over the 30 datasets. For each technique, the numbers on the main line represent its average rank. The critical difference (CD) was computed using the Bonferroni-Dunn post-hoc test. Techniques with no statistical difference are connected by additional lines

One interesting fact is that all techniques proposed in this work obtained lower rank values when compared to the previous version of the META-DES framework. The META-DES.Oracle using the V-shaped transfer function obtained the best overall performance, achieving an average rank of 3.00. Moreover, the results obtained by this technique were also significantly better than those obtained by both the META-DES and META-DES.H.

The second statistical analysis is conducted in a pairwise fashion in order to verify whether the difference in classification accuracy obtained by the META-DES.Oracle significantly improves the classification accuracy when compared to the previous versions of the framework. To that end, the Wilcoxon non-parametric signed rank test with the level of significance $\alpha = 0.05$ was used since it was suggested in [91] as a robust method for a pairwise comparison between classification algorithms over several datasets. The results of the Wilcoxon statistical test are shown in the last row of Table 4.3. Techniques that achieve performances equivalent to the META-DES.H are marked with "~"; those that achieve statistically superior performance are marked with a "+", and those with inferior performance are marked with a "-". ρ-values are also shown in the last row of Table 4.3.

The results of the Wilcoxon signed rank test also demonstrate that the META-DES.Oracle using the V-Shaped transfer function and the global validation overfitting control scheme obtained classification results that are significantly superior when compared to both the META-DES.H and the META-DES, with a 95% confidence over the 30 datasets considered in this work. Thus, based on the analysis, we can answer the two research questions posed at the beginning of this section: The meta-features selection optimization process does indeed significantly improve the classification performance of the system, when compared to the previous versions of the framework. In addition, we can also see that the system using 15 sets of meta-features without meta-feature selection, META-DES.ALL, achieves similar results when compared to previous versions of the framework (e.g., META-DES and META-DES.H). This suggest that simply adding more meta-features does not always lead to a better classification accuracy. The meta-feature selection stage is important for better addressing the behavior of the Oracle.

For the sake of simplicity, we refer to META-DES.Oracle, the version of the framework using the V-Shaped transfer function and global validation, in the rest of this paper.

4.6.5 Comparison with the state-of-the-art DES techniques

In this section, we compare the accuracy obtained by the proposed META-DES.Oracle against ten state-of-the-art dynamic selection techniques [1]. The goal of this analysis is to know if the performance of the proposed system is significantly superior when compared to state-of-the-art DES techniques. The dynamic selection techniques used in this analysis are: Local Classifier Accuracy (LCA) [22], Overall Local Accuracy (OLA) [22], Modified Local Accuracy (MLA) [29], K-Nearest Oracles-Eliminate (KNORA-E), K-Nearest Oracles-Union (KNORA-U) [14], K-Nearest Output Profiles (KNOP) [16], Multiple Classifier Behavior (MCB) [21], DES-PRC [18] and DCS-Rank [38]. These techniques were selected because they presented the very best results in the dynamic selection literature according to a recent survey on this topic [1].

The same pool of classifiers is used for all techniques in order to ensure a fair comparison. For all techniques, the size of the region of competence, K, was set at 7 since it achieved the best result in previous experiments [20; 2]. The results are shown in Table 4.4. For each dataset, we performed a pairwise comparison between the results obtained by the proposed META-DES.Oracle against those obtained by each state-of-the-art DES technique. The comparison was conducted using the Kruskal-Wallis non-parametric statistical test, with a 95% confidence interval. Results that are significantly better are marked with a •. In addition, the average rank of each technique, as well as the result of the sign test, are presented at the end of Table 4.4.

Figure 4.9 illustrates the average rank of each technique using the CD diagram. Similarly to Section 4.6.4, the CD was calculated using the Bonferroni-Dunn post-hoc test. The META-DES.Oracle obtained the lowest average rank, 2.73, followed by the technique based on probabilistic models, DES-PRC [18], presenting an average rank of 4.40. Hence, the performance of the META-DES.Oracle is significantly better when compared to the majority of the state-of-

Table 4.4 Mean and standard deviation results of the accuracy obtained for the proposed META-DES.Oracle and 10 state-of-the-art dynamic selection techniques. The best results are in bold. Results that are significantly better are marked with •

Database	META-DES.Oracle	KNORA-E [14]	KNORA-U [14]	DES-FA [20]	LCA [22]	OLA [22]	MLA [29]	MCB [21]	KNOP [16]	DES-PRC [18]	DCS-Rank [38]
Pima	**77.53(2.24)**	73.79(1.86)	76.60(2.18)	73.95(1.61)	73.95(2.98)	73.95(2.56)	77.08(4.56)	76.56(3.71)	73.42(2.11)	75.41(2.73)	72.97(2.25)
Liver Disorders	**72.02(4.72)** •	56.65(3.28)	56.97(3.76)	61.62(3.81)	58.13(4.01)	58.13(3.27)	58.00(4.25)	58.00(4.25)	65.23(2.29)	63.70(4.14)	61.24(5.42)
Breast (WDBC)	96.71(0.86)	97.59(1.10)	97.18(1.02)	**97.88(0.78)**	97.88(1.58)	97.88(1.58)	95.77(2.38)	97.18(1.38)	95.42(0.89)	96.71(0.61)	96.01(1.00)
Blood Transfusion	**79.38(1.76)** •	77.65(3.62)	77.12(3.36)	73.40(1.16)	75.00(2.87)	75.00(2.36)	76.06(2.68)	73.40(4.19)	77.54(2.03)	75.89(1.41)	74.35(2.49)
Banana	94.54(1.16)	93.08(1.67)	92.28(2.87)	**95.21(3.18)**	95.21(2.15)	95.21(2.15)	80.31(7.20)	88.29(3.38)	90.73(3.45)	86.44(1.76)	93.44(1.73)
Vehicle	82.87(1.64)	83.01(1.54)	82.54(1.70)	82.54(4.05)	80.33(1.84)	81.50(3.24)	74.05(6.65)	**84.90(2.01)**	80.09(1.47)	82.76(1.81)	79.61(1.97)
Lithuanian Classes	94.97(2.00)	93.33(2.50)	95.33(2.64)	98.00(2.46)	85.71(2.20)	**98.66(3.85)**	88.33(3.89)	86.00(3.33)	89.33(2.29)	85.04(1.57)	93.41(1.22)
Sonar	**81.63(3.90)** •	74.95(2.79)	76.69(1.94)	78.52(3.86)	76.51(2.06)	74.52(1.54)	76.91(3.20)	76.56(2.58)	75.72(2.82)	80.13(5.09)	79.27(5.67)
Ionosphere	**89.94(1.97)**	89.77(3.07)	87.50(1.67)	88.63(2.12)	88.00(1.98)	88.63(1.98)	81.81(2.52)	87.50(2.15)	85.71(5.52)	87.88(2.48)	88.51(2.87)
Wine	**99.52(1.11)** •	97.77(1.53)	97.77(1.62)	95.55(1.77)	85.71(2.25)	88.88(3.02)	88.88(3.02)	97.77(1.62)	95.50(4.14)	98.52(1.57)	92.10(5.57)
Haberman	74.22(2.85)	71.23(4.16)	73.68(2.27)	72.36(2.41)	70.16(3.56)	69.73(4.17)	73.68(3.61)	67.10(7.65)	75.00(3.40)	75.15(2.50)	70.32(4.06)
Cardiotocography (CTG)	**86.37(1.10)**	86.27(1.57)	85.71(2.20)	86.27(1.57)	86.65(2.35)	86.65(2.35)	86.27(1.78)	85.71(2.21)	86.02(3.04)	84.90(1.02)	84.98(0.84)
Vertebral Column	84.90(5.33)	85.89(2.27)	**87.17(2.24)**	82.05(3.20)	85.00(3.25)	85.89(3.74)	77.94(5.80)	84.61(3.95)	86.98(3.21)	85.90(3.68)	83.62(3.38)
Steel Plate Faults	**69.32(1.18)**	67.35(2.01)	67.96(1.98)	68.17(1.59)	66.00(1.69)	66.52(1.65)	67.76(1.54)	68.17(1.59)	68.57(1.85)	67.58(0.95)	66.55(1.64)
WDG V1	**84.72(0.49)** •	84.01(1.10)	84.01(1.10)	84.01(1.10)	80.50(0.56)	80.50(0.56)	79.95(0.85)	78.75(1.35)	84.21(0.45)	84.46(0.48)	83.85(0.61)
Ecoli	**81.57(3.47)** •	76.47(2.76)	75.29(3.41)	75.29(3.41)	75.29(3.41)	75.29(3.41)	76.47(3.06)	76.47(3.06)	80.00(4.25)	78.82(3.58)	76.73(3.52)
Glass	66.46(4.22)	57.65(5.85)	61.00(2.88)	55.32(4.98)	59.45(2.65)	57.60(3.65)	57.60(3.65)	**67.92(3.24)**	62.45(3.65)	64.99(4.23)	56.81(6.15)
ILPD	**69.79(3.15)**	67.12(2.35)	69.17(1.58)	67.12(2.35)	69.86(2.20)	69.86(2.20)	69.86(2.20)	68.49(3.27)	68.49(3.27)	67.88(1.89)	67.81(2.52)
Adult	**87.74(2.04)** •	80.34(1.57)	79.76(2.26)	80.34(1.57)	83.58(2.32)	82.08(2.42)	80.34(1.32)	78.61(3.32)	79.76(2.26)	86.71(1.53)	83.04(2.42)
Weaning	81.73(3.14)	78.94(1.25)	81.57(3.65)	**82.89(3.52)**	77.63(2.35)	77.63(2.35)	80.26(1.52)	81.57(2.86)	82.57(3.33)	78.51(3.29)	77.19(2.18)
Laryngeal1	**87.42(2.98)** •	77.35(4.45)	77.35(4.45)	77.35(4.45)	77.35(4.45)	77.35(4.45)	75.47(5.55)	77.35(4.45)	77.35(4.45)	82.18(3.79)	79.45(3.46)
Laryngeal3	**73.67(2.14)**	70.78(3.68)	72.03(1.89)	72.03(1.89)	72.90(2.30)	71.91(1.01)	61.79(7.80)	71.91(1.01)	73.03(1.89)	72.41(1.87)	66.67(6.13)
Thyroid	**96.99(0.75)** •	95.95(1.25)	95.95(1.25)	95.37(2.02)	95.95(1.25)	95.95(1.25)	94.79(2.30)	95.95(1.25)	95.95(1.25)	96.85(0.96)	96.40(1.15)
German credit	**76.58(1.99)** •	72.80(1.95)	72.40(1.80)	74.00(3.30)	73.33(2.85)	71.20(2.52)	71.20(2.52)	73.60(3.30)	73.60(3.30)	75.07(2.36)	69.78(2.70)
Heart	86.44(3.38)	83.82(4.05)	83.82(4.05)	83.82(4.05)	85.29(3.69)	85.29(3.69)	**86.76(5.50)**	83.82(4.05)	83.82(4.05)	83.66(3.64)	79.74(4.34)
Satimage	**96.65(0.83)** •	95.35(1.23)	95.86(1.07)	93.00(2.90)	95.00(1.40)	94.14(1.07)	93.28(2.10)	95.86(1.07)	95.86(1.07)	95.60(0.75)	94.76(0.97)
Phoneme	**85.05(1.08)** •	79.06(2.50)	78.92(3.33)	79.06(2.50)	78.84(2.53)	78.84(2.53)	64.94(7.75)	73.37(5.55)	78.92(3.33)	73.64(1.55)	79.45(0.88)
Monk2	**94.45(1.88)** •	80.55(3.32)	77.77(4.25)	75.92(4.25)	74.07(6.60)	74.07(6.60)	75.92(5.65)	74.07(6.60)	80.55(3.32)	80.86(2.58)	86.21(4.93)
Mammographic	80.72(2.56)	82.21(2.27)	82.21(2.27)	80.28(3.02)	82.21(2.27	82.21(2.27)	75.55(5.50)	81.25(2.07)	82.21(2.27)	**84.29(1.32)** •	79.75(3.48)
MAGIC Gamma Telescope	86.02(2.20)	80.03(3.25)	79.99(3.55)	81.73(3.27)	81.53(3.35)	81.16(3.00)	73.13(6.35)	75.91(5.35)	80.03(3.25)	**86.20(1.52)**	76.72(1.13)
Average rank	**2.73(1.34)**	5.56(1.29)	5.33(1.07)	5.90(1.54)	6.20(1.44)	6.80(1.44)	7.30(1.52)	7.33(1.45)	5.93(1.51)	4.40(1.61)	8.10(1.53)
Win-tie-loss	n/a	4-3-23	6-1-23	6-1-23	5-2-23	7-1-22	3-2-25	5-1-24	5-0-25	4-2-24	1-0-29
Wilcoxon Signed Test	n/a	− ($p = .0003$)	− ($p = .0014$)	− ($p = .0014$)	− ($p = .0152$)	− ($p = .0161$)	− ($p = .0001$)	− ($p = .0003$)	− ($p = .0003$)	− ($p = .0003$)	− ($p = .0003$)

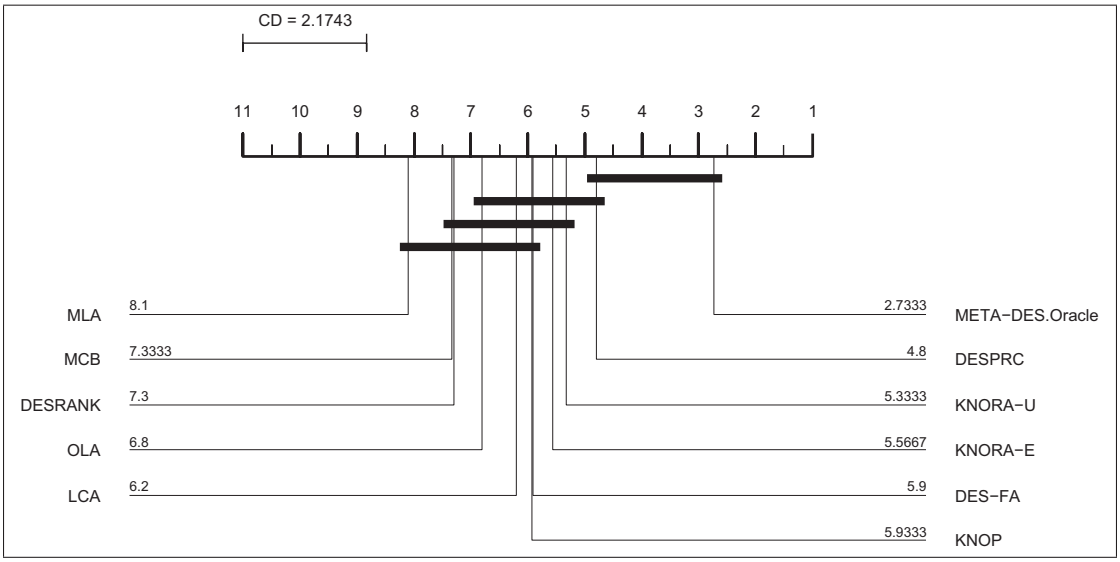


Figure 4.9 Average rank of the dynamic selection methods over the 30 datasets. The best algorithm is the one presenting the lowest average rank

the-art DES techniques. Only the DES-PRC obtained a statistically equivalent performance. However, when we compared those two techniques in terms of wins, ties and losses as reported in Table 4.4, we could see that the META-DES.Oracle obtained the best accuracy for

24 datasets, while the DES-PRC outperformed the META-DES.Oracle only in 4 datasets. For two datasets, the results of both techniques were tied. Furthermore, we also performed the Wilcoxon non-parametric signed rank test with the level of significance $\alpha = 0.05$ for a pairwise comparison between the results obtained by the META-DES.Oracle against state-of-the-art DES techniques over the 30 datasets. The results of the Wilcoxon test are presented in the last row of Table 4.4.

When a pairwise comparison between the techniques is performed, we can see that the META-DES.Oracle dominates when compared against previous DES techniques. Its performance is statistically better when compared to any of the 10 state-of-the-art techniques. This can be explained by two factors: state-of-the-art DES techniques are based only on one criterion to estimate the competence of the base classifier; this could be, local accuracy, ranking, probabilistic models, etc. For instance, the ranking and probabilistic criteria used by the DCS-RANK and DES-PRC techniques are embedded in the META-DES framework as meta-features f_{rank} and f_{PRC}, respectively. In addition, through the BPSO meta-features selection scheme, only the meta-features that are relevant for the given classification problem are selected and used for the training of the meta-classifier. As shown in Figure 4.6, the selected meta-features vary considerably according to different classification problems. Thus, it is expected that the proposed framework obtains a significant gain in performance when compared to previous DES techniques.

4.6.6 Comparison with Static techniques

In this section, we compare the results obtained by the META-DES.Oracle against static ensemble techniques as well as single classifier models. For the static ensemble techniques, we evaluate the performance of the AdaBoost [5], Bagging [3], the classifier with the highest accuracy in the validation data (Single Best) and a static ensemble selection method based on the majority voting error proposed in [66]. Furthermore, three single classifier models are considered: MLP Neural Network, Support Vector Machines with Gaussian Kernel (SVM) and Random Forest classifier. These classifiers were selected based on a recent study [71]

that ranked the best classifiers in a comparison considering a total of 179 classifiers over 121 classification datasets.

The objective of this study is to determine whether the proposed META-DES.Oracle obtain recognition accuracy that is either statistically better or equivalent to the ones achieved by the best classifiers in the literature [71]. This is an important analysis since the DES literature still lacks a comparison with classical classification approaches that do not use ensembles. In the DES literature, the accuracy of the proposed techniques are only compared either with other DES techniques or with static ensemble selection considering the same pool of classifiers [1].

All classifiers were evaluated using the Matlab PRTOOLS toolbox [63]. Since static techniques require neither a meta-training nor a dynamic selection phase, the training ($\mathcal{T}$) and meta-training set ($\mathcal{T}_\lambda$) were merged into a single training set. The dynamic selection dataset (DSEL) was used as the validation dataset. The test set, $\mathcal{G}$, remained unchanged. For each replication, the parameters of the single classifier model were set as follows:

a. MLP Neural Network: We varied the number of neurons in the hidden layer from 10 to 100 at 10 point intervals. The configuration that achieved the best results in the validation data was used. The MLP training process was conducted using the Levenberg-Marquadt algorithm. The process was stopped if the performance on the validation set decreased or failed to improve for five consecutive epochs.

b. SVM with a Gaussian Kernel: A grid search was performed in order to set the values of the regularization parameter, c, and the Kernel spread parameter γ.

c. Random Forest: We varied the number of trees from 25 to 200 at 25 point intervals. The configuration with the highest performance on the validation dataset was used for generalization.

The classification accuracy of each technique is reported in Table 4.5. For each dataset, we performed a pairwise comparison between the results obtained by the proposed META-

DES.Oracle, against the results obtained by each state-of-the-art DES technique. The comparison was conducted using the Kruskal-Wallis non-parametric statistical test, with a 95% confidence interval. Results that are significantly better at a 95% confidence are marked with •. Moreover, we also report the average ranks and the results of the Wilcoxon test at the end of Table 4.5. Figure 4.10 illustrates the critical difference diagram.

Table 4.5 Mean and standard deviation results of the accuracy obtained for the proposed META-DES and static classification models. The best results are in bold. Results that are significantly better ($p < 0.05$) are marked with •

Database	META-DES.Oracle	Single Best [1]	Bagging [3]	AdaBoost [5]	Static Selection [66]	MLP NN	SVM	Random Forest
Pima	**77.53(2.24)**	73.57(1.49)	73.28(2.08)	72.52(2.48)	72.86(4.78)	69.37(2.94)	76.56(2.71)	74.32(3.92)
Liver Disorders	**72.02(4.72)**	65.38(3.47)	62.76(4.81)	64.65(3.26)	59.18(7.02)	61.86(4.86)	71.27(4.10)	67.32(4.79)
Breast (WDBC)	96.71(0.86)	97.04(0.74)	96.35(1.14)	**98.24(0.89)** •	96.83(1.00)	95.77(0.74)	97.81(1.07)	95.85(1.37)
Blood Transfusion	**79.38(1.76)** •	75.07(1.83)	75.24(1.67)	75.18(2.08)	75.74(2.23)	76.38(1.48)	75.42(4.23)	73.03(6.35)
Banana	94.54(1.16)	84.07(2.22)	81.43(3.92)	81.61(2.42)	81.35(4.28)	98.11(0.85)	**98.19(0.78)** •	97.02(1.03)
Vehicle	**82.87(1.64)** •	81.87(1.47)	82.18(1.31)	80.56(4.51)	81.65(1.48)	72.31(8.63)	74.19(3.00)	79.00(2.42)
Lithuanian Classes	94.97(2.00)	84.35(2.04)	82.33(4.81)	82.70(4.55)	82.66(2.45)	92.66(3.15)	**96.40(1.70)** •	95.53(1.50)
Sonar	81.63(3.90)	78.21(2.36)	76.66(2.36)	74.95(5.21)	79.03(6.50)	76.15(6.09)	82.80(3.99)	**84.80(6.62)** •
Ionosphere	89.94(1.97)	87.29(2.28)	86.75(2.75)	86.75(2.34)	87.50(2.23)	86.36(4.31)	**94.54(1.58)** •	94.09(2.50)
Wine	**99.52(1.11)** •	96.70(1.46)	95.56(1.96)	99.20(0.76)	96.88(1.80)	92.88(10.30)	98.88(1.17)	97.33(2.29)
Haberman	74.52(2.94)	**75.65(2.68)**	72.63(3.45)	75.26(3.38)	73.15(3.68)	68.42(5.15)	71.10(2.21)	63.81(7.23)
Cardiotocography (CTG)	86.37(1.10)	84.21(1.10)	84.54(1.46)	83.06(1.23)	84.04(2.02)	88.19(2.27)	**92.29(0.76)** •	91.27(1.20)
Vertebral Column	84.90(5.33)	82.04(2.17)	85.89(3.47)	83.22(3.59)	84.27(3.24)	84.14(4.55)	84.74(4.33)	84.48(3.93)
Steel Plate Faults	69.32(1.18)	66.05(1.98)	67.02(1.98)	66.57(1.06)	67.22(1.64)	68.99(2.63)	**74.00(1.72)** •	69.83(3.05)
WDG V1	84.72(0.49)	83.17(0.76)	84.36(0.56)	84.04(0.37)	84.23(0.53)	81.68(7.82)	**86.90(0.09)** •	85.89(0.46)
Ecoli	81.57(3.47)	69.35(2.68)	72.22(3.65)	70.32(3.65)	67.80(4.60)	74.35(14.08)	**83.88(2.42)**	67.65(7.55)
Glass	66.46(4.22)	52.92(4.53)	62.64(5.61)	55.89(3.25)	57.16(4.17)	56.22(7.99)	60.60(5.17)	**66.54(6.01)**
ILPD	**69.79(3.15)**	67.53(2.83)	67.20(2.35)	69.38(4.28)	67.26(1.04)	64.31(3.68)	66.23(3.95)	65.68(3.94)
Adult	**87.74(2.04)** •	83.64(3.34)	85.60(2.27)	83.58(2.91)	84.37(2.79)	80.33(3.25)	85.31(3.06)	83.03(4.60)
Weaning	81.73(3.14)	74.86(4.78)	76.31(4.06)	74.47(3.68)	76.89(3.15)	80.92(4.77)	87.23(1.96)	**88.25(2.93)** •
Laryngeal1	**87.42(2.98)** •	80.18(5.51)	81.32(3.82)	79.81(3.88)	80.75(4.93)	76.98(6.01)	81.69(4.70)	80.18(4.81)
Laryngeal3	73.67(2.14)	68.42(3.24)	67.13(2.47)	62.32(2.57)	71.23(3.18)	64.26(4.19)	**74.60(2.95)**	71.12(4.73)
Thyroid	**96.99(0.75)** •	95.15(1.74)	95.25(1.11)	96.01(0.74)	96.24(1.25)	94.98(1.35)	94.79(0.10)	95.08(0.49)
German credit	**76.58(1.99)** •	71.16(2.39)	74.76(2.73)	72.96(1.25)	73.60(2.69)	64.20(3.98)	75.32(1.70)	70.35(5.85)
Heart	**86.44(3.38)** •	80.26(3.58)	82.50(4.60)	81.61(5.01)	82.05(3.72)	71.17(6.86)	83.44(3.28)	77.79(3.27)
Satimage	**96.65(0.83)**	94.52(0.96)	95.23(0.87)	95.43(0.92)	95.31(0.92)	92.65(2.97)	91.15(1.20)	96.21(1.42)
Phoneme	85.05(1.08)	75.87(1.33)	72.60(2.33)	75.90(1.06)	72.70(2.32)	82.11(4.17)	76.27(1.85)	**89.59(0.20)** •
Monk2	94.45(1.88)	79.25(3.78)	79.18(2.57)	80.27(2.76)	80.55(3.59)	**99.25(1.21)** •	96.57(1.38)	83.88(3.09)
Mammographic	80.72(2.56)	83.60(1.85)	**85.27(1.85)** •	83.07(3.03)	84.23(2.14)	77.88(9.87)	80.29(1.83)	77(1.12)
MAGIC Gamma Telescope	86.02(2.20)	80.27(3.50)	81.24(2.22)	**87.35(1.45)**	85.25(3.25)	83.07(2.20)	87.20(1.52)	**88.65(2.32)**
Average rank	**2.43(0.86)**	5.40(0.87)	4.80(1.02)	5.26(1.03)	4.70(0.83)	4.93(1.16)	3.26(1.04)	4.20(1.28)
Win-Tie-Loss	n/a	3-1-26	4-0-26	5-1-24	3-1-26	4-1-25	12-3-15	10-2-18
Wilcoxon Signed Test	n/a	- ($\rho = .0001$)	- ($\rho = .0001$)	- ($\rho = .0003$)	- ($\rho = .0001$)	- ($\rho = .0101$)	~ ($\rho = .3600$)	~ ($\rho = .2005$)

Based on the result,s we can conclude that the META-DES.Oracle outperforms static ensemble techniques. This result was expected since many works in the DES literature have shown that dynamic selection outperforms static combination rules in many applications [1]. Moreover, this claim is especially true when a pool of weak linear classifiers is considered since they become experts into different regions of the feature space. As reported in [37], a static combination of base classifiers in such a case may not yield a good classification performance since there may never be a consensus in the correct answer between the classifiers in the pool. However, when dynamic selection is used, only the most competent classifiers for the given

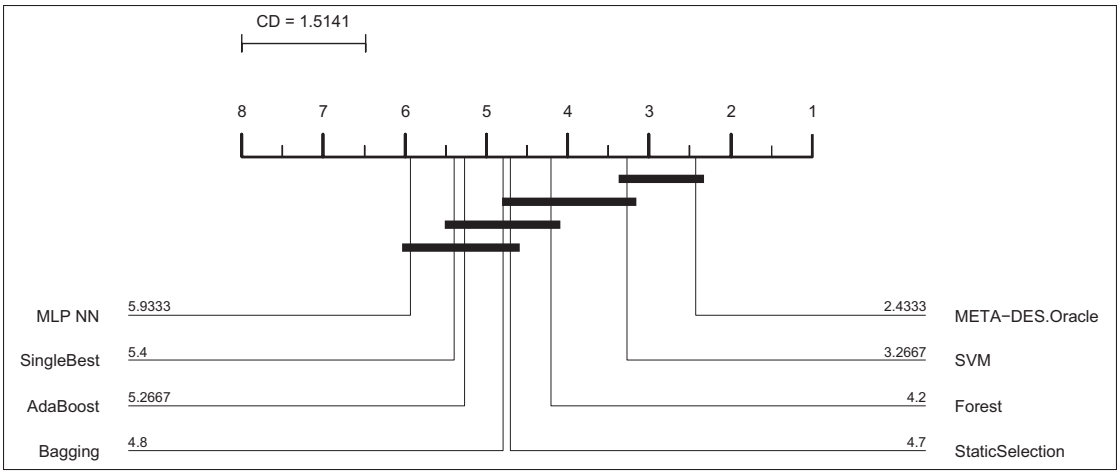


Figure 4.10 Average rank of the dynamic selection methods over the 30 datasets. The best algorithm is the one presenting the lowest average rank

query sample are selected to predict its label. As such, the classifiers that are not experts in the local region do not influence the ensemble decision negatively.

When compared with single classifier models, the META-DES.Oracle obtained the lowest average rank. The results achieved META-DES.Oracle is statistically equivalent to those achieved by the SVM classifier, based on both the Friedman test with Bonferroni-Dunn post-hoc test, and the Wilcoxon sign test at $\alpha = 0.05$ significance. Hence, the analysis demonstrate the classification performance achieved by the proposed META-DES.Oracle is among the best classifier models in the literature, since both SVM and Random Forests presented the overall best performance in the analysis conducted by Delgado et al. [71].

It is important to point out that the META-DES.Oracle obtained a small advantage in terms of wins, ties and losses when compared to the SVM classifier. The META-DES.Oracle presented the best recognition accuracy in 16 datasets, while the SVM obtained a higher accuracy in 12 datasets. For two datasets (Vertebral Column and Mammographic), the results were tied. This result may be explained by the fact most of the datasets used in this analysis are ill-defined, i.e., small sample size datasets. For such datasets, the training data may not have enough samples to train a single classifier model and select the best hyperparameters, e.g., the number of neurons in the hidden layer of an MLP neural network, or the regularization

parameter, c, and the Kernel spread parameter, γ, in an SVM. In addition, since the training set was small, there might be variations between the training and test distribution. The META-DES.Oracle obtained the best results for several ill-defined problems, such as Liver disorders, Blood transfusion, Heart, Laryngeal1, Wine and Thyroid. Those are all small-sized datasets with less than 500 samples available for training. One advantage of the META-DES framework is that the pool is composed of linear classifiers which do not require the selection of any hyper-parameters. Thus, the training can be performed using small size datasets. Since the training set is relatively small, the classifiers may specialize in local regions of the feature space. Using dynamic selection, only the most competent classifiers in the local region where the test sample is located are used to predict its label. Thus, through DES, it is still possible to obtain high classification accuracy even for ill-defined problems.

Furthermore, the optimization process of the META-DES.Oracle framework is conducted in the meta-problem, using the meta-data extracted in the meta-training stage. Several meta-feature vectors are generated for each training sample in the meta-training phase. For instance, consider that 200 training samples are available for the meta-training stage ($N = 200$); if the pool C is composed of 100 weak classifiers ($M = 100$), the meta-training dataset is the number of training samples $N \times$ the number classifiers in the pool M, $N \times M = 20.000$. Hence, even though the problem may be ill-defined, the framework generates enough meta-training data in order to properly train the meta-classifier. There is more data to train the meta-classifier λ than for the generation of the pool of classifiers C itself. Hence, even though the classification problem may be ill-defined, given the size of the training set, using the proposed framework, we can overcome this limitation since the size of the meta-problem is up to 100 times bigger than the classification problem.

4.7 Conclusion

In this chapter, we propose a novel DES framework using meta-learning and Oracle information, called META-DES.Oracle. 15 sets of meta-features are proposed, using different sources of information found in the DES literature for dynamically estimating the level of competence

of base classifiers; these include, local accuracy, ranking, probabilistic, ambiguity and behavior. Next, a meta-feature selection scheme using overfitting cautious Binary Particle Swarm Optimization is performed to optimize the performance of the meta-classifier. The optimization process is guided by a formal definition of the Oracle. Thus, the meta-classifier can better address the complex behavior of the Oracle.

We have conducted a case study using the P2 problem, which is a synthetic dataset with a complex non-linear decision border. We demonstrate that using a pool composed of 5 linear Perceptron classifiers, it is possible to approximate the complex decision boundary of the P2 problem using the proposed framework. The proposed META-DES.Oracle obtained a recognition performance of 97%, which is closer to the results obtained by the Oracle, and compares very favorably against previous versions of the META-DES framework.

Experiments were conducted using 30 classification problems. First, we performed an analysis of the meta-features that were selected for each problem. The analysis demonstrated that the selected sets of meta-features varies considerably according to different datasets. In addition, each meta-feature was selected in at least 20% of the datasets. All sets of meta-features was thus relevant in better addressing the complex behavior of the Oracle. Next, the performance obtained by the proposed META-DES.Oracle was compared with previous versions of the META-DES framework, as well as ten state-of-the-art dynamic selection techniques. Experimental results demonstrate that the META-DES.Oracle outperforms the previous versions of the technique in the majority of the datasets. In addition, the gain in performance obtained by the META-DES.Oracle is shown to be statistically significant based on both the Friedman test with a post-hoc Bonferroni-Dunn correction and the Wilcoxon sign rank test. Thus, the BPSO meta-features selection scheme proposed in this paper does indeed significantly improve the classification performance of the framework.

When compared with static and single classifier methods, the results achieved by the proposed META-DES.Oracle are comparable with the best performing classifiers. Moreover, the results confirm the claim that DES techniques outperform single classifier models for ill-defined prob-

lems. Since the optimization process of the META-DES.Oracle is performed using the meta-data generated during the meta-training stage, there is enough data to train and optimize the meta-classifier. Thus, the proposed framework can deal with small sample size classification problems.

CHAPTER 5

PROTOTYPE SELECTION FOR DYNAMIC CLASSIFIER AND ENSEMBLE SELECTION

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article submitted to « Neural Computing and Applications » 2016.

Abstract

In dynamic ensemble selection (DES) techniques, only the most competent classifiers, for the classification of a specific test sample are selected to predict the sample's class labels. The key in DES techniques is estimating the competence of the base classifiers for the classification of each specific test sample. The classifiers' competence is usually estimated according to a given criterion, which is computed over the neighborhood of the test sample defined on the validation data, called the region of competence. A problem arises when there is a high degree of noise in the validation data, causing the samples belonging to the region of competence to not represent the query sample. In such cases, the dynamic selection technique might select the base classifier that overfitted the local region rather than the one with the best generalization performance. In this paper, we propose two modifications in order to improve the generalization performance of any DES technique. First, a prototype selection technique is applied over the validation data to reduce the amount of overlap between the classes, producing smoother decision borders. During generalization, a local adaptive K-Nearest Neighbor algorithm is used to minimize the influence of noisy samples in the region of competence. Thus, DES techniques can better estimate the classifiers' competence. Experiments are conducted using 10 state-of-the-art DES techniques over 30 classification problems. The results demonstrate that the proposed scheme significantly improves the classification accuracy of dynamic selection techniques.

5.1 Introduction

In the last few years, dynamic ensemble selection (DES) [1] has become an active research topic in multiple classifier systems. The rationale behind such techniques resides in the observation that not every classifier in the pool is an expert in classifying all unknown samples. Each base classifier[1] is an expert in a different local region of the feature space [27].

Dynamic selection techniques consist, based on a pool of classifiers C, in finding a single classifier c_i, or an ensemble of classifiers C', that has (or have) the most competent classifiers to predict the label for a specific test sample, $\mathbf{x}_j$. The most important component of DES techniques is how the competence level of the base classifier is measured, given a specific test sample $\mathbf{x}_j$. Usually, the competence of a base classifier is estimated based on instances that are similar to the query instance, using the K-Nearest Neighbors (KNN) technique and a set of labeled samples, which can be either the training or validation set. In this paper, we refer to such a set as the dynamic selection dataset (DSEL), following the conventions of the dynamic selection literature [2; 1]. The set with the K-Nearest Neighbors of a given test sample $\mathbf{x}_j$ is called the region of competence, and is denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. The samples belonging to θ_j are used to estimate the competence of the base classifiers, for the classification of $\mathbf{x}_j$, based on various criteria, such as the overall accuracy of the base classifier in this region [22], ranking [38], ambiguity [15], oracle [14] and probabilistic models [45].

A problem arises with dynamic selection techniques when the samples in the local region are not representative enough of the query sample. This may be seen in cases in which a high degree of overlap is present between the classes, and as a result of noise or outliers. As reported in [37], the performance of dynamic selection techniques is very sensitive to the distribution of DSEL.

In order to illustrate how the presence of noise in DSEL can lead to poor classification results by using a dynamic selection technique, we perform a case study using the synthetic P2 problem proposed in [92]. The P2 is a bi-dimensional two-class synthetic classification prob-

[1]The term base classifier refers to a single classifier belonging to an ensemble or a pool of classifiers

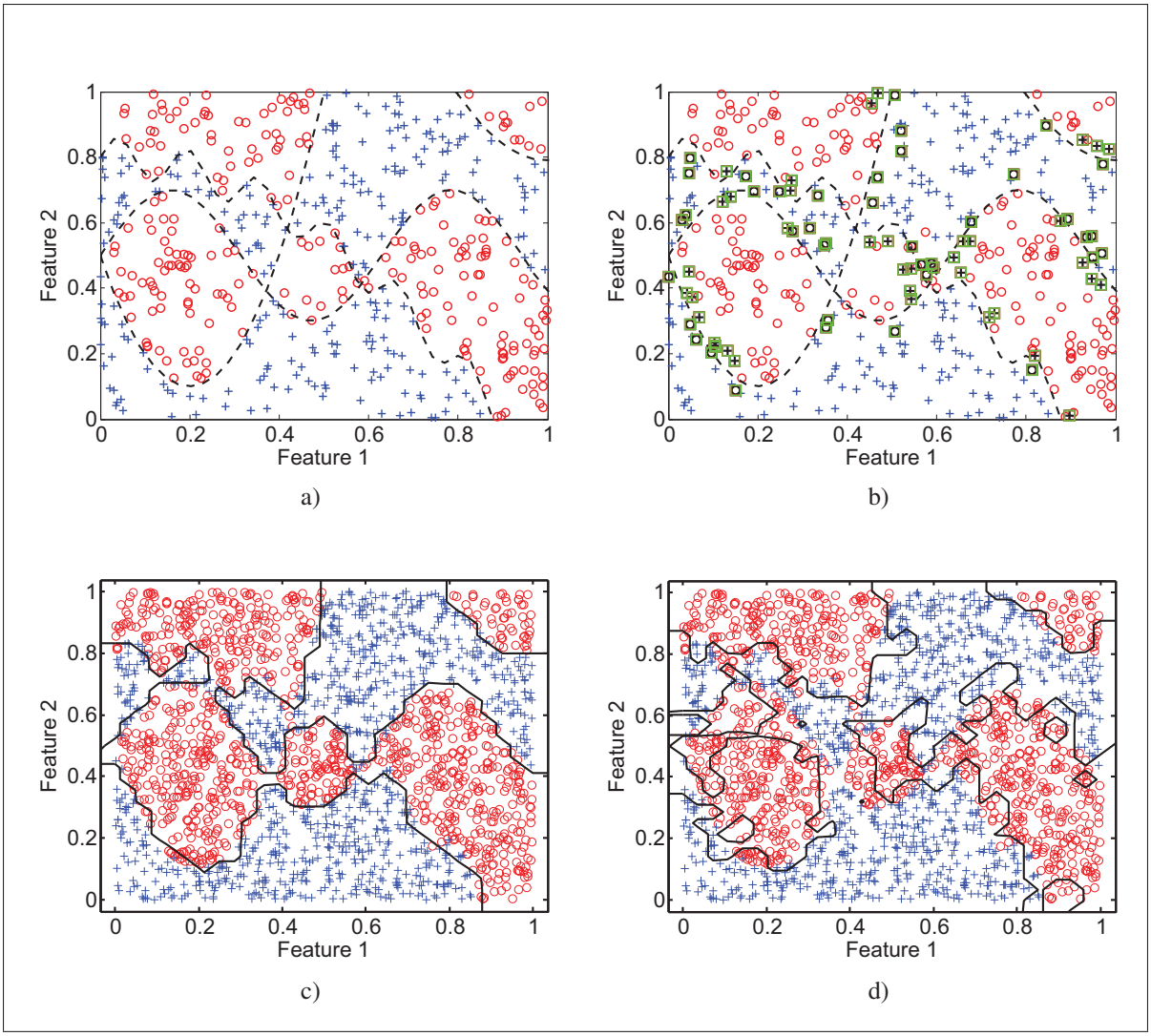


Figure 5.1 Case study using the synthetic P2 problem. The red circle represents the class 1 and the blue cross the class 2. The axes represent the values of the two features of the P2 problem. (a) The original distribution of DSEL. (b) The distribution of DSEL with 25% of added noise by switching labels of samples close to the class borders. The noisy samples are highlighted in green. (c) Result of the META-DES framework using the Original DSEL. (d) Results of the META-DES framework using the noisy DSEL.

lem in which each class is defined in multiple decision regions delimited by polynomial and trigonometric functions.

For this example, the META-DES framework proposed in [2] is considered since it outperformed several dynamic selection techniques in multiple classification benchmarks. The P2

problem was generated using the same methodology reported in [37]: 500 samples for the training set ($\mathcal{T}$), 500 instances for the dynamic selection dataset, DSEL, and 2000 samples for the test set, $\mathcal{G}$. The original distribution of DSEL is shown in Figure 5.1 (a). The red circle and blue cross represent samples belonging to class 1 and class 2, respectively.

Since there is no overlap between the classes in the original distribution, we generate noise in DSEL by switching the labels of samples that are close to the decision border with a 25% probability (Figure 5.1 (b)). Samples that had their class labels changed are highlighted in green. Figures 5.1 (c) and (d) show the approximation of the P2 border achieved by the META-DES framework plotted over the test distribution. Figure 5.1 (c) presents the decision achieved using the original distribution of DSEL. In contrast, Figure 5.1 (d) presents the decision obtained using DSEL with 25% of added noise. We can observe that the META-DES fails to obtain a good approximation of the decision boundary of the P2 Problem when noise is added to DSEL. Moreover, the errors committed by the META-DES occur in regions of the feature space where the presence of noise in DSEL is more evident.

This work therefore aims to improve the classification accuracy of dynamic selection techniques by reducing the presence of noise in DSEL. The proposed scheme is based on two steps: The first modification proposed in this paper applies a prototype selection mechanism to the dynamic selection set, DSEL, in order to eliminate instances highly likely to be noise, and also reduces the amount of overlap between the classes. The Edited Nearest Neighbor (ENN) [93] rule is used for this purpose. Secondly, the local regions of the query sample are estimated using an adaptive KNN rule (AKNN), which shifts the region of competence from the class border to the class centers. Samples that are more likely to be noise are less likely to be selected to compose the region of competence. As such, we expect the dynamic selection technique to be able to better estimate the competence level of a base classifier, leading to better generalization performance. It should be mentioned that the proposed method can be applied to any dynamic selection technique that uses local information in estimating the competence of the base classifier.

The proposed approach is evaluated using 10 state-of-the-art dynamic classifier and ensemble selection techniques over 30 classification datasets. We evaluate four scenarios: (I) The dynamic selection techniques using the original dynamic selection dataset and the standard KNN algorithm for computing the region of competence θ_j; (II) The ENN is applied to edit DSEL and the standard KNN is used; (III) Only the AKNN technique is used, and (IV) Both the ENN and the AKNN techniques are used. The following research questions are analyzed: (1) Does the prototype selection technique lead to an improvement in classification accuracy? (2) Which scenario produces the best recognition rates? (3) Which dynamic selection technique benefits the most from the proposed scheme?

This paper is organized as follows: The proposed approach is detailed in Section 5.2. The experimental study is conducted in Section 5.3. Finally, our conclusion and future works are presented in the last section.

5.2 Proposed method

Two changes are proposed in this paper; one during the training stage, and the other in the generalization stage. In the training stage, we apply a prototype selection technique in the dataset DSEL in order to remove noise and outliers. To that end, the Edited Nearest Neighbor technique is considered since it is able to significantly reduce the presence of noise in the dataset, thereby improving the KNN performance [93]. During the generalization stage, given a new test sample $\mathbf{x}_{j,test}$, the region of competence θ_j is computed based on the samples in the edited dynamic selection dataset, denoted by DSEL$'$, using a local adaptive distance rule. Both techniques are presented in the following sections.

5.2.1 Edited Nearest Neighbor (ENN)

There are three types of prototype selection mechanisms available [94]: condensation, edition and hybrid. Condensation techniques are used in order to reduce the dataset size, without losing the generalization performance of the system. Edition techniques aim to improve the

performance of the KNN algorithm by removing instances with a high risk of being noise. The editing process occurs in regions of the feature space with a high degree of overlap between classes, producing smoother class boundaries. Hybrid techniques perform both a condensation of the data and edition of the class borders. Since our goal is to improve the classification accuracies, an edition technique is performed. The Edited Nearest Neighbor (ENN) [93] is used since it is a very well-known technique for removing noise and decreasing the amount of overlap in the class borders, producing smoother decision borders. Moreover, the ENN technique is known to significantly improve the performance of the KNN [94].

Input: Dynamic Selection Dataset DSEL
1: $\text{DSEL}' = \text{DSEL}$
2: **for** each $\mathbf{x}_{j,DSEL} \in \text{DSEL}$ **do**
3: **if** $label\left(\mathbf{x}_{j,DSEL}\right) \neq label\left(KNN\left(\mathbf{x}_{j,DSEL}\right)\right)$ **then**
4: $\text{DSEL}' = \text{DSEL}' \setminus \left\{\mathbf{x}_{j,DSEL}\right\}$
5: **end if**
6: **end for**
7: **return** DSEL'

Algorithm 5.1: The Edited Nearest Neighbor rule

Given the dynamic selection dataset DSEL, the ENN algorithm works as follows (Algorithm 5.1): For each instance $\mathbf{x}_{j,DSEL} \in$ DSEL, the class label of $\mathbf{x}_{j,DSEL}$ is predicted using the KNN algorithm using a leave-one-out procedure. A K = 3 was used, as suggested by Wilson [93], in order to satisfy the asymptotic properties of the NN technique. If $\mathbf{x}_{j,DSEL}$ is misclassified by the KNN technique, it is removed from the set, since $\mathbf{x}_{j,DSEL}$ is in a region of the feature space where the majority of samples belongs to a different class. The edited dynamic selection dataset, denoted by DSEL', is obtained at the end of the process.

It should be mentioned that the ENN does not remove all samples in the class borders, and that the intrinsic geometry of the class borders and the distribution of the classes are preserved. Only instances that are associated with a high degree of instance hardness, i.e., those for which the majority of neighbors belong to a different class, are removed. As reported in [95], these samples have a reputation for being hard to be correctly classified by the majority of learning

algorithms. Only the classifiers that overfitted the training data are able to predict its correct class label. In such cases, the dynamic selection technique might select the base classifier that overfitted the local region rather than the one that has the best generalization performance in the region. By removing these instances, we expect the dynamic selection techniques to be able to better estimate the base classifier's competences.

5.2.2 K-Nearest Neighbor with Local Adaptive Distance

The locally adaptive distance for KNN was proposed in [96]. For each sample $\mathbf{x}_{j,DSEL'}$ in the edited dynamic selection dataset DSEL$'$, the largest hypersphere centered on $\mathbf{x}_{j,DSEL'}$, which excludes all samples in DSEL$'$ with a different class label, is constructed (Figure 5.2). Such a hypersphere is built by computing its radius $R_{j,DSEL'}$, which is measured as the minimum distance between the sample $R_{j,DSEL'}$ and a sample from a different class $\mathbf{x}_{jk,DSEL'}$ (Equation 5.1):

$$R_{j,DSEL'} = d\left(\mathbf{x}_{j,DSEL'}, \mathbf{x}_{jk,DSEL'}\right) - \varepsilon \tag{5.1}$$

where $d\left(\mathbf{x}_{j,DSEL'}, \mathbf{x}_{jk,DSEL'}\right)$ is the Euclidean distance between the instances $\mathbf{x}_{j,DSEL'}$ and $\mathbf{x}_{jk,DSEL'}$, ε is a small number (in this work $\varepsilon = 0.01$). $w_{j,DSEL'}$ and $w_{jk,DSEL'}$ are the labels of $\mathbf{x}_{j,DSEL'}$ and $\mathbf{x}_{jk,DSEL'}$, respectively.

Each instance belonging to DSEL$'$ is associated with a hypersphere of radius $R_{j,DSEL'}$. The hypersphere associated with each sample delimits the region within which its class label can be generalized to other samples without making an error [96]. The hypersphere associated with samples that are closer to the class center have a larger radius since they are more distant from samples from different classes, when compared to those hyperspheres that are associated with samples that are closer to the class boundaries. Figure 5.2 illustrates an example of the hypersphere associated with different samples from the P2 problem.

The adaptive distance between a given test sample $\mathbf{x}_{j,test}$ and a sample belonging to $DSEL'$, $\mathbf{x}_{j,DSEL'}$, is obtained using Equation 5.2.

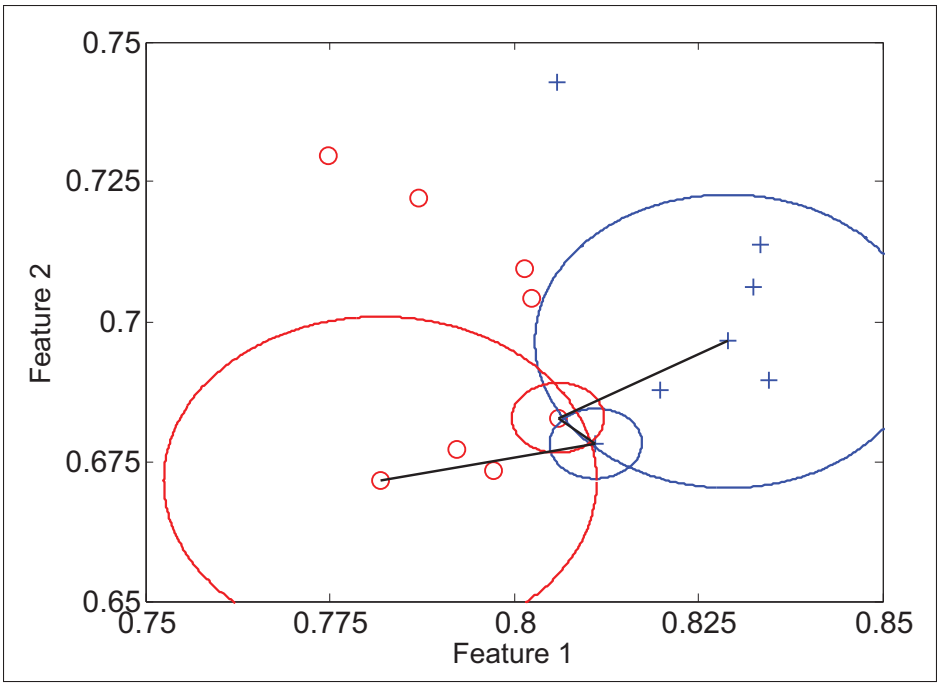


Figure 5.2 Example of the hypersphere associated with the samples in DSEL′, considering the P2 problem. The red circle and the blue cross represent samples belonging to class 1 and class 2, respectively. The X- and Y-axes indicate the values of the two features of the P2 problem.

$$d_{adaptive}(\mathbf{x}_{j,test}, \mathbf{x}_{j,DSEL'}) = \frac{d(\mathbf{x}_{j,test}, \mathbf{x}_{j,DSEL'})}{R_{j,DSEL'}} \tag{5.2}$$

The distance is said to be adaptive since the influence of each sample is normalized by a factor $R_{j,DSEL'}$, which changes according to the spatial location of each instance in DSEL′. The larger the value of $R_{j,DSEL'}$ (i.e., larger hypersphere), the lower the value of $d_{adaptive}$. The A-KNN technique is beneficial in regions where there is a high degree of overlap between the two classes, since it tends to identify samples that have larger hyperspheres as the nearest neighbors to the query sample. As reported in [96], the majority of K-Nearest Neighbors selected are more likely to have the same class label as the query sample. Thus, the dynamic selection algorithm can better estimate the competence of the base classifiers for the classification of $\mathbf{x}_{j,test}$.

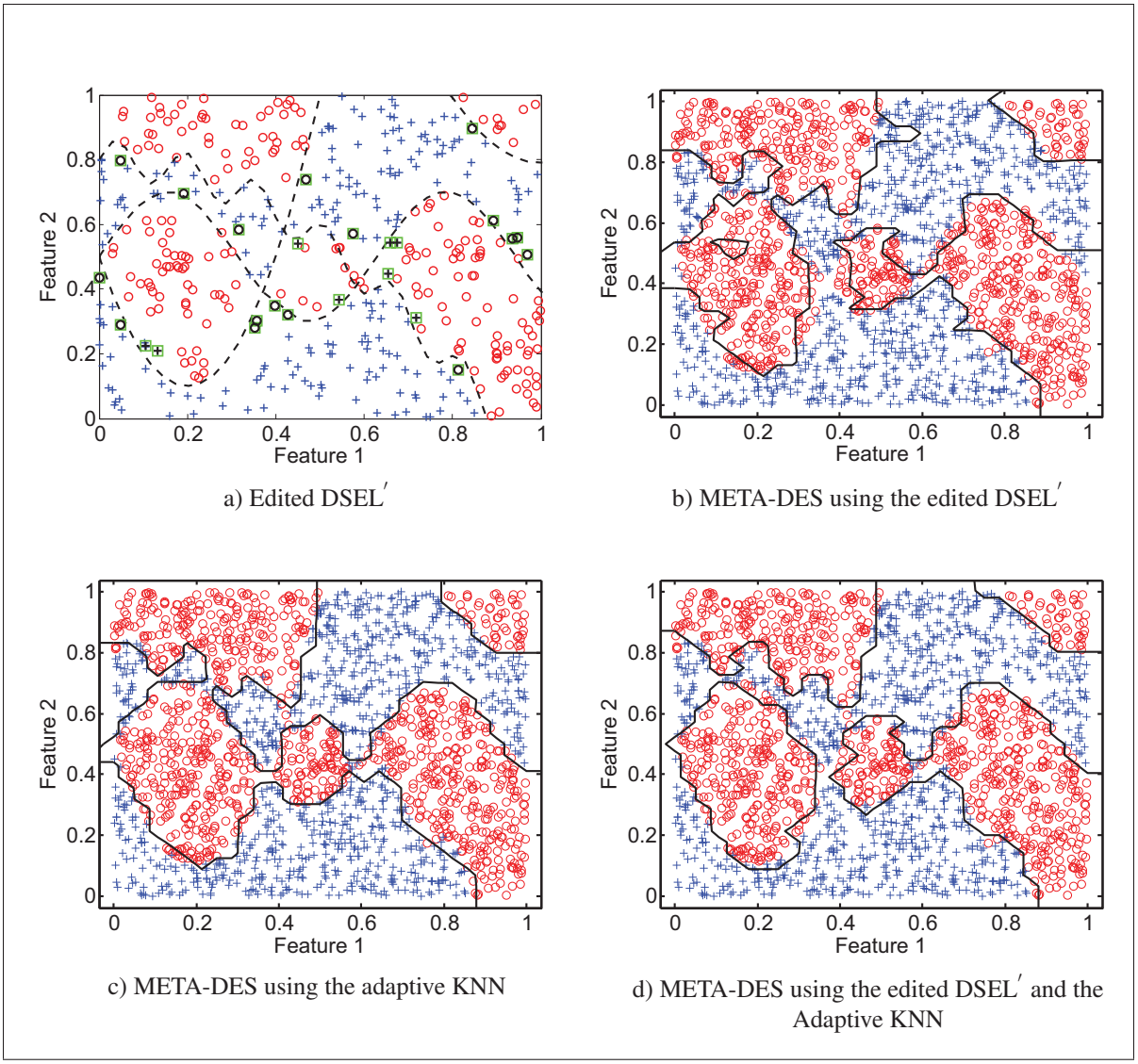


Figure 5.3 Case study using the two-dimensional P2 problem. The axes represent the values of the two features of the P2 problem: (a) Distribution of DSEL after applying the ENN technique to clean the border. Noisy samples are highlighted in green; (b) Result of the META-DES framework using DSEL′ for computing the local regions; (c) Result of the META-DES using the adaptive distance (AKNN); (d) Result of the META-DES framework using both the ENN and the AKNN techniques.

5.2.3 Case study

Using the same distributions of the P2 problem discussed in Section 1, if we apply the ENN technique in editing the dynamic selection dataset, the overlap in the decision boundary is

significantly removed. Figure 5.3 (a) shows the distribution of the edited dynamic selection dataset $DSEL^{'}$ using the ENN prototype selection technique. Noisy samples are highlighted in green. We can see that the majority of noisy samples were removed from DSEL. In addition, we can see that the geometry of the decision border is still preserved. Figure 5.3 (b) shows the result of the META-DES technique using the $DSEL^{'}$ in computing the local regions. The META-DES can have a closer approximation of the real decision boundary of the P2 problem. However, it can be seen that there are still some outliers in the edited DSEL, and their presence still negatively affects the performance of the system.

The adaptive distance comes in handy in those cases since there is no guarantee that the ENN will completely remove all noisy samples from DSEL. If we also use the adaptive distance (Figure 5.3 (c)) in computing the region of competence θ_j, the META-DES can obtain a decision boundary that is close to those obtained using a noise-free DSEL. Thus, by editing the dynamic selection dataset and the adaptive KNN distance, we can obtain a good approximation of the decision boundary of the P2 problem, even with a high noise presence.

5.3 Experiments

In this section, we compare the impact of the adaptive distance and the editing of the class boundaries using several state-of-the-art dynamic classifier selection and dynamic ensemble selection techniques found in the literature.

5.3.1 Dynamic selection methods

A total of 10 dynamic selection techniques were considered in the experiments. In order to have a balance between dynamic classifier selection (DCS) and dynamic ensemble selection (DES), we considered five techniques from each paradigm. In addition, based on the dynamic selection taxonomy proposed in [1], there were five categories: Ranking, Accuracy, Oracle, Probabilistic and Behavior. To ensure the availability of a diverse set of techniques, we considered at least one technique taken from each category. We also included the META-DES in the

experimental study which was published after the survey, and could be considered as belonging to a different category (meta-learning). Thus, methods that use different sources of information for estimating the competence level of the base classifiers were considered in the experimental study. Table 5.1 illustrates the 10 dynamic selection techniques considered in this work.

For dynamic classifier selection, the following techniques were considered: Local classifier Accuracy (LCA) [22], Overall Local Accuracy (OLA) [22], Modified Local Accuracy (MLA) [29], Classifier ranking (RANK) [38] and the Multiple Classifier Behavior (MCB) [21]. The following techniques for dynamic ensemble selection were considered: K-Nearest Oracles Eliminate (KNORA-E) [14], K-Nearest Oracles Union (KNORA-U) [16], Randomized Reference Classifier (DES-PRC) [40], K-Nearest Output Profiles (KNOP) [16; 17], and the META-DES framework [2]. The pseudo-code for each technique can be found in the following survey [1], and in [2], for the META-DES framework.

Table 5.1 Dynamic selection techniques considered in the experiments. Pseudo-code for each technique can be found in the following survey [1], and in [2], for the META-DES framework.

	Technique	**Category**	**Reference**
DCS	Classifier Rank (RANK)	Ranking	Sabourin et al. [38]
	Local Classifier Accuracy (LCA)	Accuracy	Woods et al.[22]
	Overall Local Accuracy (OLA)	Accuracy	Woods et al.[22]
	Modified Local Accuracy (MLA)	Accuracy	P.C. Smits[29]
	Multiple Classifier Behavior (MCB)	Behavior	Giacinto et al.[21]
DES	K-Nearests Oracles Eliminate (KNORA-E)	Oracle	Ko et al.[14]
	K-Nearests Oracles Union (KNORA-U)	Oracle	Ko et al.[14]
	Randomized Reference Classifier (RRC)	Probabilistic	Woloszynski et al.[18]
	K-Nearests Output Profiles (KNOP)	Behavior	Cavalin et al.[16]
	META-DES	Meta-Learning	Cruz et al.[2]

5.3.2 Datasets

The experiments were conducted on 30 datasets taken from five different data repositories. Sixteen datasets were taken from the UCI machine learning repository [59], four from the STATLOG project [60], four from the Knowledge Extraction based on Evolutionary Learning

(KEEL) repository [61], four from the Ludmila Kuncheva Collection of real medical data [62], and two artificial datasets generated with the Matlab PRTOOLS toolbox [63]. The experimental study is focused on small size datasets, since, as reported by Cavalin et al. [16], dynamic selection techniques have been shown to be an effective tool for problems where the level of uncertainty for recognition is high due to few training samples being available. However, a few larger datasets, such as the Magic gamma telescope, phoneme and Adult, were also considered in order to evaluate the performance of the proposed scheme for different types of classification problems.

Since ensemble methods have recently become popular in dealing with the class imbalance problem [97; 98], several imbalanced datasets, such as Ecoli, Glass, Satimage and Phoneme, were also considered. Table 5.2 presents the main characteristics of the 30 classification datasets. The imbalanced ratio (IR) is measured by the number of instances of the majority class per instance of the minority class. Thus, a higher IR value indicates a higher degree of imbalance.

In order to ensure a fair comparison between the results obtained by the proposed technique and those from the DES literature, the same experimental setup as in previous works [2] is considered. For each dataset, the experiments were carried out using 20 replications. For each replication, the datasets were randomly divided on the basis of 50% for training, $\mathcal{T}$, 25% for the dynamic selection dataset, $DSEL$, and 25% for the generalization set, $\mathcal{G}$. The divisions were performed while maintaining the prior probabilities of each class. Since the META-DES framework requires an additional training step for the training of the meta-classifiers (meta-training), 25% of the training set was used in the meta-training phase. The pool of classifiers C was composed of 100 Perceptrons generated using the Bagging technique. The size of the region of competence (neighborhood size) K was equally set at 7 for all techniques. The hyper-parameters for the META-DES framework were set according to guidelines proposed by the authors [69; 37].

Table 5.2 Summary of the 30 datasets used in the experiments [Adapted from [2]]. The imbalanced ratio (IR) is measured by the number of instances of the majority class per instance of the minority class.

Database	No. of Instances	Dimensionality	No. of Classes	IR	Source
Pima	768	8	2	1.87	UCI
Liver Disorders	345	6	2	1.37	UCI
Breast (WDBC)	568	30	2	1.86	UCI
Blood transfusion	748	4	2	3.20	UCI
Banana	1000	2	2	1.00	PRTOOLS
Vehicle	846	18	4	1.09	STATLOG
Lithuanian	1000	2	2	1.00	PRTOOLS
Sonar	208	60	2	1.14	UCI
Ionosphere	315	34	2	1.78	UCI
Wine	178	13	3	1.47	UCI
Haberman's Survival	306	3	2	2.78	UCI
Cardiotocography (CTG)	2126	21	3	9.40	UCI
Vertebral Column	310	6	2	2.1	UCI
Steel Plate Faults	1941	27	7	14.05	UCI
WDG V1	5000	21	3	1.02	UCI
Ecoli	336	7	8	71.50	UCI
Glass	214	9	6	8.44	UCI
ILPD	583	10	2	2.49	UCI
Adult	48842	14	2	3.17	UCI
Weaning	302	17	2	1.00	LKC
Laryngeal1	213	16	2	1.62	LKC
Laryngeal3	353	16	3	4.19	LKC
Thyroid	215	5	3	12.05	LKC
German credit	1000	20	2	2.33	STATLOG
Heart	270	13	2	1.25	STATLOG
Satimage	6435	19	7	9.29	STATLOG
Phoneme	5404	6	2	2.41	ELENA
Monk2	4322	6	2	1.11	KEEL
Mammographic	961	5	2	1.05	KEEL
MAGIC Gamma Telescope	19020	10	2	1.84	KEEL

5.3.3 Comparison between different scenarios

We evaluated four different scenarios for the dynamic selection techniques (Table 5.3).

For each scenario, we evaluated each dynamic selection technique over the 30 datasets, for a total of 300 experiments (30 datasets $\times$ 10 techniques) per scenario. To compare the four approaches, the Friedman rank analysis was conducted since it is a robust statistical method for comparing multiple techniques over several datasets [91].

Table 5.3 Four test scenarios

Scenario	ENN	Adaptive KNN
I	No	No
II	Yes	No
III	No	Yes
IV	Yes	Yes

For each dataset and dynamic selection method, the Friedman test ranks each scenario, with the best performing one getting rank 1, the second best rank 2, and so forth. Then, the average rank of each scenario is calculated. The best scenario is the one that obtained the lowest average rank. After the average ranks were computed, the post-hoc Bonferroni-Dunn test was conducted for a pairwise comparison between the ranks achieved by each scenario. The performance of two techniques is significantly different if their difference in average rank is higher than the critical difference (CD) calculated by the Bonferroni-Dunn post-hoc test. The average ranks of the four scenarios, as well as the results of the post-hoc test, are presented using the CD diagram [91] (Figure 5.4). We can see, based on the CD diagram, that the performance of Scenario IV is statistically better when compared to the other scenarios.

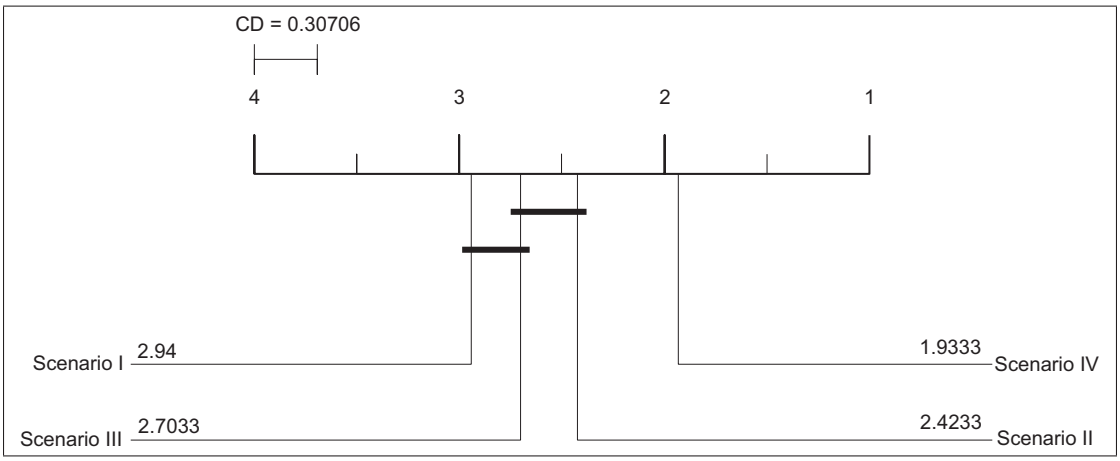


Figure 5.4 Critical difference diagram considering the four test scenarios. The best algorithm is the one presenting the lowest average rank. Techniques that are statistically equivalent are connected by a black bar.

In addition to the Friedman analysis, we also conducted a pairwise comparison between Scenario I (without using the ENN and A-KNN) and the other test scenarios, using the sign test [91] calculated on the computed wins, ties and losses. The null hypothesis H_0 meant that both approaches yielded equivalent results, and a rejection in H_0 meant that the proposed approach was significantly better at a predefined significance level. In this work, we use the significance level $\alpha = 0.05$. To reject H_0, the number of wins needs to be greater than or equal to the critical value n_c calculated using Equation 5.3:

$$n_c = \frac{n_{exp}}{2} + z_\alpha \frac{\sqrt{n_{exp}}}{2} \tag{5.3}$$

where n_{exp} is the total number of experiments (10 techniques $\times$ 30 datasets = 300), and $z_\alpha = 1.645$, for a significance level of $\alpha = 0.05$. Hence, $n_c = 170.14$.

Considering Scenario IV, the number of wins, ties and losses are 195, 23 and 82, respectively. However, for computing the test, half the ties are added to the wins and the other half to the losses, which gives us 206.5 wins and 93.5 losses. H_0 is rejected since $206.5 > 170.14$. Scenario II also presented a significant gain in performance, with 186 wins and 114 losses, while the performance of Scenario III was statistically equivalent (152 wins and 148 losses).

Based on the statistical analysis, we can conclude that Scenarios II and IV achieve results that are statistically better when compared to Scenario I. Thus, the proposed scheme does indeed lead to significant gains in performance for dynamic selection techniques. We can also observe that the editing of DSEL using the ENN technique is the main factor in improving the classification performance, since Scenario II also presented a significant gain in performance when compared to Scenario I, while the performance of Scenario I and III was statistically equivalent.

5.3.4 Comparison between DES techniques

In order to identify which dynamic selection technique benefited the most from the proposed scheme, we conducted an analysis considering each technique separately. We performed a pairwise comparison between each DES technique using Scenarios I and IV. Only Scenario IV is considered in this analysis since it outperformed Scenarios II and III in the previous experiment. The comparison was conducted using the sign test calculated on the computed wins, ties and losses. The null hypothesis, H_0, meant that the corresponding DES technique achieved equivalent results using Scenarios I and IV. In this case, the total number of experiments for each DES technique is equal to the number of datasets $n_{exp} = 30$.

In order to reject H_0 at $\alpha = 0.05$, the number of wins plus half the number of ties achieved by a dynamic selection technique must be greater than or equal to the critical value, $n_c = 19.5$. As shown in Figure 5.5, the META-DES, OLA, LCA, KNORA-E, DCS-RANK and DES-PRC achieved significant performance gains using the proposed approach.

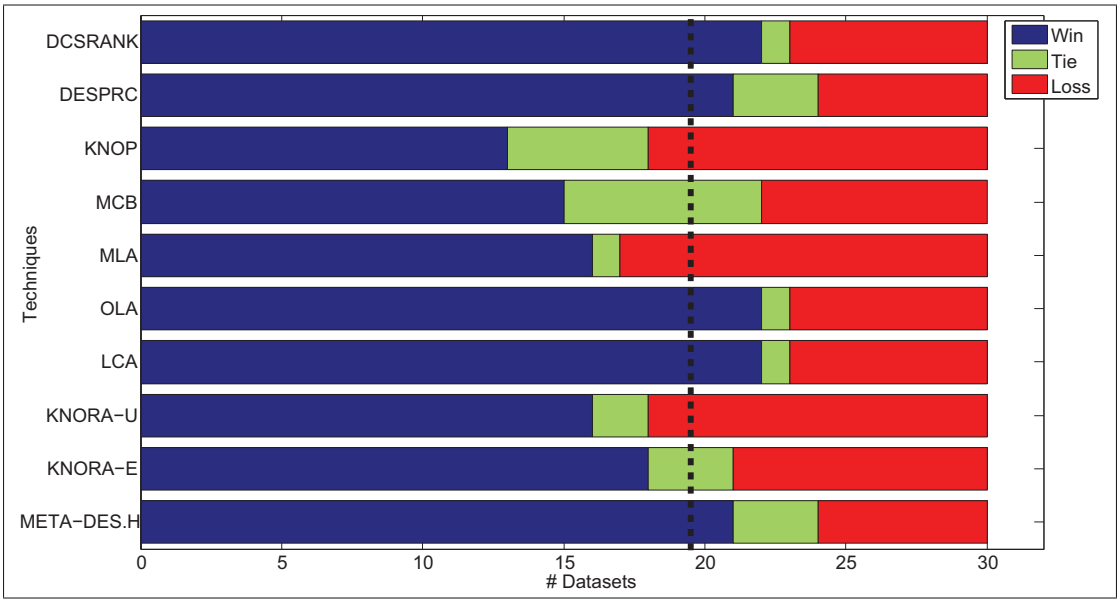


Figure 5.5 Performance of the each dynamic selection technique using the ENN and A-KNN in terms of wins, ties and losses. The dashed line illustrates the critical value $n_c = 19.5$.

Furthermore, the Friedman test was used in order to compare the results of all the DES techniques over the 30 classification datasets (Figure 5.6), using Scenarios I and IV. Techniques marked with an * are the ones using the ENN and A-KNN (Scenario IV). It can be seen that all DES techniques presented a lower average rank when using the proposed scheme (Scenario IV). Moreover, the techniques that are based purely on local accuracy information, such as LCA and OLA and DCS-RANK, presented a greater benefit, i.e., difference between the average ranks. For instance, the LCA* achieved an average rank of 9.96, while the average rank for the original LCA technique was 12.96. Techniques that are not based on the information extracted from the feature space, such as the MCB, which estimates the competence of the base classifier using the decision space, are the ones with smaller differences in average ranks (12.0 obtained by MCB against 11.4 achieved by the MCB*), which may simply be explained by the fact the ENN technique reduces the amount of overlap in the feature space rather than the decision space. Since the META-DES technique obtained the lowest average rank, we also present the classification accuracies obtained by the META-DES and META-DES* for the 30 classification datasets (Table 5.4). The best results are highlighted in bold.

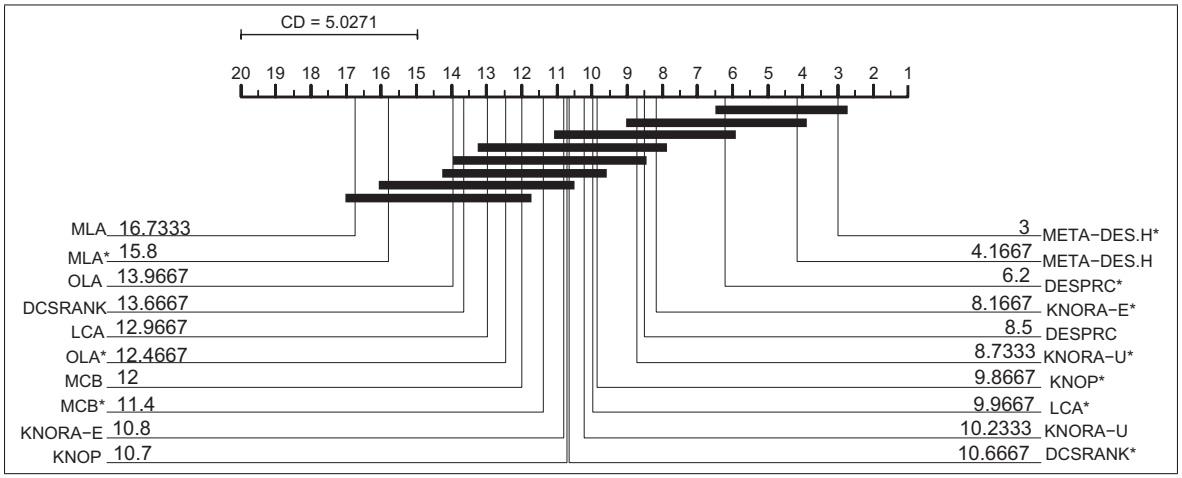


Figure 5.6 CD diagram considering all techniques. Techniques marked with a * are the ones using Scenario IV.

5.3.5 Discussion

Looking at the classification results in Table 5.4, we can see that the proposed scheme works well when dealing with problems with few classes, even when considering datasets with a high degree of overlap between them, such as the Liver, Blood and Monk2, datasets. The proposed scheme failed to improve the classification accuracy only in a few datasets. These datasets generally have the same characteristics: They are both heavily imbalanced and small-sized. In such cases, there may not be enough samples in the dynamic selection dataset for the ENN filter and the AKNN to work properly. In fact, the ENN technique tends to remove instances from the minority class since they are under-represented, and some isolated instances may be considered as noise by the algorithm. Hence, we believe that the best strategy to deal with problems that are heavily imbalanced involves using a prototype generation technique, such as in [99; 100], to generate samples for the minority class, and apply the prototype selection only for the majority class.

Another important aspect of the proposed scheme is that, by removing samples in DSEL, the running time of the dynamic selection techniques decreases. For every technique, the running time to classify a given test instance $\mathbf{x}_j$ of each method is a combination of the definition of the region of competence and evaluating the competence level of each classifier in the pool. The definition of the region of competence is performed only once as it depends only on the input sample $\mathbf{x}_j$, and not on the base classifier. Since it is performed based on the AKNN technique, the cost is of order $O(d \times N)$, given that d and N are the number of dimensions and samples in the dynamic selection dataset (DSEL$'$), respectively.

For each dynamic selection technique, the outputs of the base classifiers for the samples in DSEL must first be pre-calculated during the training stage of the system and stored in a matrix. The storage requirement for the pre-calculated information is $O(M \times N \times \Omega)$, with M and Ω being the number of classifiers in the pool and the number of classes in the dataset. The computational cost involved during generalization consists in accessing the outputs of the base classifier stored in the matrix and applying the selection criteria for each base classifier in the

pool. Thus, the cost of evaluating the competence of each classifier in the pool of classifiers is $O(M)$.

Therefore, besides improving the classification accuracy, the proposed scheme can also reduce the memory requirement and the running time of dynamic selection techniques during the generalization phase.

5.4 Conclusion

In this paper, we demonstrate that the performance of DES techniques is sensitive to the dynamic selection dataset distribution. A high degree of overlap in the dynamic selection dataset may lead to poor estimations of the local competence of the base classifiers; thus, the dynamic selection technique fails to select the most appropriate classifier for the classification of a new query sample. We show that with two simple modifications, we can significantly improve the generalization performance of any dynamic selection technique.

In order to evaluate the impact of the proposed scheme, we compared the results of ten dynamic classifier selection and dynamic ensemble selection techniques over 30 classification datasets. The experimental results demonstrate that the proposed scheme significantly improves the classification accuracy of the dynamic selection techniques. The scenario using both the ENN and A-KNN techniques presented the overall best result. In addition, using only the ENN for editing the dynamic selection dataset brings about a significant gain in classification accuracy.

Future work will include the evaluation of different prototype selection techniques, as well as prototype generation for dealing with problems that are both small sized and heavily imbalanced.

Table 5.4 Comparison of the results achieved by META-DES framework [2], considering Scenarios I and IV. Best results are highlighted.

Dataset	META-DES.H*	META-DES.H
Pima	**78.80(3.23)**	77.93(1.86)
Liver	**70.73(4.10)**	69.69(4.68)
Breast	97.02(0.49)	**97.25(0.47)**
Blood	**79.85(1.12)**	78.25(1.37)
Banana	**95.51(1.46)**	94.51(2.36)
Vehicle	83.24(2.05)	**83.55(2.10)**
Lithuanian	**94.75(2.71)**	93.26(3.22)
Sonar	**82.06(5.74)**	82.06(5.09)
Ionosphere	**89.31(2.94)**	89.06(2.21)
Wine	**99.02(1.83)**	98.53(1.48)
Haberman	75.83(1.58)	**76.13(2.06)**
CTG	**86.89(1.41)**	86.08(1.24)
Vertebral	**87.47(7.05)**	84.90(2.95)
Faults	**69.41(1.36)**	68.95(1.04)
WDVG1	84.63(0.75)	**84.77(0.65)**
Ecoli	**80.66(3.48)**	**80.66(3.48)**
GLASS	**65.21(3.53)**	**65.21(3.53)**
ILPD	**70.02(2.82)**	69.64(2.47)
Adult	**87.74(2.84)**	87.29(1.80)
Weaning	**81.15(3.33)**	79.98(3.55)
Laryngeal1	**87.63(4.19)**	87.21(5.35)
Laryngeal3	**74.17(2.89)**	73.54(0.67)
Thyroid	96.99(6.13)	**97.38(1.66)**
German	**75.87(2.59)**	74.36(1.28)
Heart	**85.62(3.34)**	85.46(2.70)
Segmentation	**96.52(1.06)**	96.46(0.79)
Phoneme	**82.68(1.31)**	81.82(0.69)
Monk2	**92.40(2.58)**	83.45(3.46)
Mammographic	80.24(8.61)	**84.30(2.27)**
Magic	**86.02(2.20)**	85.65(2.27)

GENERAL CONCLUSION

In this thesis, a dynamic selection framework using meta-learning, called META-DES, is proposed. The framework is based on two environments: the classification environment, in which the input features are mapped into a set of class labels, and the meta-classification environment, in which different properties from the classification environment, such as the classifier accuracy in the feature space or the consensus in the decision space, are extracted from the training data and encoded as meta-features. Several sets of meta-features are proposed based on distinct sources of information to characterize the competence of the base classifier, for the classification of a specific test sample, such as local accuracy and confidence. These meta-features are used to train a meta-classifier which can estimate whether or not a base classifier is competent enough to classify a given input sample. With the arrival of new test data, the meta-features are extracted using the test data as reference, and used as input to the meta-classifier. The meta-classifier decides whether the base classifier is competent enough to classify the test sample.

In Chapter II, the dynamic ensemble selection is formalized as a meta-problem. Then, a novel DES framework using meta-learning, called META-DES, is proposed. In addition, five sets of meta-features for the given meta-problem are proposed based on different DES criteria. The proposed META-DES obtained higher classification performance when compared to several state-of-the-art dynamic classifier and ensemble selection techniques.

In Chapter III, a deep analysis of the META-DES framework is conducted in order to understand why it succeeds in achieving high recognition performance using only a few linear classifiers. The analysis is conducted using the P2 problem, which is a complex non-linear problem with two multi modal classes. The influence of each set of meta-features on the selection of the most competent classifier is analyzed, as well as other aspects of the framework, such as the influence of the sample selection mechanism, the size of the pool of classifiers, and the distribution of the dynamic selection dataset (DSEL). Moreover, we show that the META-DES framework can approximate the complex decision boundary of the P2 problem using either Perceptrons or Decision Stumps as base classifiers, while static combination techniques such

as AdaBoost and Majority Voting fail to approximate the complex decision boundary of the P2 problem using such classifiers. The lessons learned in this analysis are used as a guideline for further improvements not only for the META-DES framework, but for DES techniques in general.

In Chapter IV, an evolution of the META-DES framework, called META-DES.Oracle, is presented. First, 10 new sets of meta-features are proposed in order to explore different sources of information for dynamically estimating the competence level of the base classifiers, such as probabilistic models, entropy and ranking. Then, a meta-feature selection scheme using an overfitting cautious BPSO is proposed for optimizing the performance of the meta-classifier. The BPSO optimization is conducted based on the formal definition of the Oracle. The differences between the outputs of the meta-classifier and the ideal outputs estimated by the Oracle are minimized. Two topologies of the BPSO technique are considered, namely, V-shaped and S-shaped. Experimental results show that the proposed optimization scheme significantly improves the classification performance of the META-DES framework. In addition, the results also demonstrate that the selection of the best sets of meta-features is also problem-dependent. The meta-classifier requires different sets of meta-features in order to achieve a performance close to those of the Oracle for different classification problems.

Lastly, in Chapter V, we show that the performance of the META-DES framework can be sensitive to the presence of noise in the dynamic selection dataset (DSEL). Two techniques ware suggested in order to improve classification accuracy in the presence of noise. During the training stage, the Edited Nearest Neighbor prototype selection technique is applied over DSEL for eliminating noise and reducing the amount of overlap among the classes. During the dynamic selection (generalization) stage, the Adaptive KNN distance is used, and so samples that are more likely to be noise have a smaller chance of being selected to compose the region of competence. The proposed scheme is applied to ten dynamic classifier and ensemble selection techniques, including the proposed META-DES. Classification results demonstrate that the generalization performance of dynamic selection techniques using both ENN and A-KNN significantly improves the classification accuracy of several DES techniques. Moreover,

the prototype selection method applied over the dynamic selection dataset (DSEL) is the main factor in improving the classification performance of several DES techniques.

Future Works

The findings of this work suggests the following points as future works in this topic:

- A classifier generation technique for use in dynamic selection. Currently in the DES literature, techniques, such as bagging and random subspace are used. However, such techniques were proposed to deal with static ensembles, and when they are used, the base classifiers are generated based on a random sampling of the training data, with no guarantee that there is diversity between the classifiers generated. In addition, some of the feature space might not be covered by any of the base classifiers. The analysis conducted in Chapter 3 shows that there are plenty of redundant classifiers in the pool when bagging is used in generating the pool of classifiers.

- The results obtained in Chapters 3 and 5 indicates that the performance of both the META-DES and the other DES techniques in the literature depends on the distribution of the dynamic selection dataset (DSEL). An interesting research direction would involve understanding the relationship between the distribution of DSEL and the decision hyperplanes of the base classifiers.

- The use of prototype generation techniques in populating areas of the feature space where the samples are sparse is another interesting research direction, since the experiments conducted in Chapters 3 and 5 demonstrate that the distribution of DSEL has a huge influence in the classification performance of dynamic selection techniques.

.

APPENDIX I

ON META-LEARNING FOR DYNAMIC ESEMBLE SELECTION

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Published in « International Conference on Pattern Recognition (ICPR) » 2014.

Abstract

In this paper, we propose a novel dynamic ensemble selection framework using meta-learning. The framework is divided into three steps. In the first step, the pool of classifiers is generated from the training data. The second phase is responsible to extract the meta-features and train the meta-classifier. Five distinct sets of meta-features are proposed, each one corresponding to a different criterion to measure the level of competence of a classifier for the classification of a given query sample. The meta-features are computed using the training data and used to train a meta-classifier that is able to predict whether or not a base classifier from the pool is competent enough to classify an input instance. Three different training scenarios for the training of the meta-classifier are considered: problem-dependent, problem-independent and hybrid. Experimental results show that the problem-dependent scenario provides the best result. In addition, the performance of the problem-dependent scenario is strongly correlated with the recognition rate of the system. A comparison with state-of-the-art techniques shows that the proposed-dependent approach outperforms current dynamic ensemble selection techniques.

1. Introduction

Ensembles of Classifiers (EoC) have been widely studied in the past years as an alternative to increase efficiency and accuracy in many pattern recognition [24; 9]. There are many examples

in the literature that show the efficiency of an ensemble of classifiers in various tasks, such as signature verification [101], handwritten recognition [11; 102] and image labeling [28]. Classifiers ensembles involve two basic approaches, namely classifier fusion and dynamic ensemble selection. With classifier fusion approaches, every classifier in the ensemble is used and their outputs are aggregated to give the final prediction. However, such techniques [24; 103; 104; 11] presents two main problems: they are based on the assumption that the base classifiers commit independent errors, which is difficult to find in real pattern recognition applications. Moreover, not every classifier in the pool of classifiers is an expert for every test pattern. Different patterns are associated with distinct degrees of difficulties. It is therefore reasonable to assume that only a few base classifiers can achieve the correct prediction.

On the other hand, dynamic ensemble selection (DES) techniques work by estimating the level of competence of a classifier for each query sample separately. Then, only the most competent classifiers in relation to the input sample are selected to form the ensemble. Thus, the key point in DES techniques is to define a criterion to measure the level of competence of a base classifier for the classification of the given query sample. In the literature, we can observe several criteria based on estimates of the classifier accuracy in local regions of the feature space surrounding the query sample [22; 20; 14; 29; 18], extent of consensus [15] and decision templates [21; 48; 16; 64]. However, in our previous works [20], we demonstrate that using only one criterion to measure the level of competence of a base classifier is very error-prone.

In this paper, we propose a novel dynamic ensemble selection framework using meta-learning. The framework is divided into three steps: (1) overproduction, where the pool of classifiers is generated, (2) Meta-training where the meta-features are extracted, using the training data, and used as inputs to train a meta-classifier that works as a classifier selector. Five sets of meta-features are proposed in this work. Each set of meta-features correspond to a different criteria used to measure the level of competence of a base classifier such as the confidence of the base classifier for the classification of the input sample, and its performance in predefined regions of the feature space. (3) Generalization phase, in which the meta-features are extracted from each query sample and used as input to the meta-classifier to perform the ensemble selection. Thus,

based on the proposed framework we integrate multiple dynamic selection criteria in order to achieve a more robust dynamic selection technique.

Three different training scenarios for the meta-classifier are investigated: (1) The meta-classifier is trained using data from one classification problem, and is used as the classifier selector [1] on the same problem; (2) The meta-classifier is trained using one classification problem, and is used as the classifier selector on a different one; (3) A single meta-classifier is trained using the data of all classification problems considered in this work, and is used as the classifier selector for all classification problems.

Based on these three scenarios, we aim to answer three research questions: (1) Can the use of meta-features lead to a more robust dynamic selection technique? (2) Is the training of the meta-classifier problem-dependent? (3) Can we improve the performance of the meta-classifier using knowledge from different classification problems? Experiments conducted over eleven classification datasets demonstrate that the proposed technique outperforms current dynamic selection techniques. Furthermore, the accuracy of the DES system is correlated to the performance of the meta-classifier.

This paper is organized as follows: In Section 2 we introduce the notion of classifier competence for dynamic selection. The architecture of the proposed system is presented in Section 3. Experimental results are given in Section 4. Finally, a conclusion is presented in the last section.

2. Classifier Competence

The level of competence of a classifier defines how much we trust an expert, given a classification task. It is used as a way of selecting, from a pool of classifiers C, the one(s) that best fit(s) a given test pattern $\mathbf{x}_j$. Thus, in dynamic selection, the level of competence is measured on-the-fly according to some criteria applied for each input instance separately. There are three

[1] In this paper, we use the terms meta-classifier and classifier selector interchangeably

categories present in the literature [1]: the classifier accuracy over a local region, i.e., in a region close to the test pattern; decision templates, and the extent of consensus.

2.1 Classifier accuracy over a local region

Classifier accuracy is the most commonly used criterion for dynamic classifier and ensemble selection techniques [22; 14; 20; 30; 38; 29; 19]. Techniques that are based on this paradigm first define a local region around the test instance, called the region of competence. This region is computed using either the K-NN algorithm [14; 22; 20] or by Clustering techniques [30; 42]. For example, the OLA technique [22] selects the classifier that obtains the highest accuracy rate in the region of competence. The Local classifier accuracy (LCA) [22] selects the classifier with the highest accuracy in relation to a specific class label and the K-Nearests Oracle (KNORA) technique [14] selects all classifiers that achieve a perfect accuracy in the region of competence. The drawback of these techniques is that their performance ends up limited by the algorithm that defines the region of competence [20].

2.2 Decision Templates

In this class of methods, the goal is also to select patterns that are close to the test sample $\mathbf{x}_j$. However, the similarity is computed in the decision space through the concept of decision templates [57]. This is performed by transforming both the test instance $\mathbf{x}_j$ and the validation data into output profiles using the transformation T, ($T : \mathbf{x}_j \Rightarrow \tilde{\mathbf{x}}_j$), where $\mathbf{x}_j \in \Re^D$ and $\tilde{\mathbf{x}}_j \in Z^M$ [17; 64] (M is the pool size). The output profile of a pattern $\mathbf{x}_j$ is denoted by $\tilde{\mathbf{x}}_j = \left\{\tilde{\mathbf{x}}_{j,1}, \tilde{\mathbf{x}}_{j,2}, \ldots, \tilde{\mathbf{x}}_{j,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,i}$ is the decision yielded by the classifier c_i for $\mathbf{x}_j$. Based on the information extracted from the decision space, the K-Nearest Output Profile (KNOP) [17] is similar to the KNORA technique, with the difference being that the KNORA works in the feature space while the KNOP works in the decision space. The Multiple Classifier Behaviour (MCB) technique [21] selects the classifiers that achieve a performance higher than a given threshold. The problem with using such information lies in the fact it neglects the local performance of the base classifiers.

2.3 Extent of Consensus or confidence

In this class of techniques, the first step is to generate a population of an ensemble of classifiers (EoC), $C^* = \{C'_1, C'_2, \ldots, C'_{M'}\}$ (M' is the number of EoC generated) using an optimization algorithm such as a genetic algorithms or greedy search [13; 12]. Then, for each new query instance $\mathbf{x}_j$, the level of competence of each EoC is computed using techniques such as the Ambiguity-guided dynamic selection (ADS), Margin-based dynamic selection (MDS) and Class-strength dynamic selection (CSDS) [15; 16]. The drawback of these techniques is that they require the pre-computation of EoC, which increases the computational complexity. In addition, the pre-computation of EoC also reduces the level of diversity and the Oracle performance (the Oracle performance is the upper limit performance of an EoC [9]) of the pool [15].

3. Proposed dynamic ensemble selector

A general overview of the proposed framework is depicted in Figure I-1. It is divided into three phases: Overproduction, Meta-training and Generalization.

3.1 Overproduction

In this step, the pool of classifiers $C = \{c_1, \ldots, c_M\}$, where M is the pool size, is generated using the training dataset $\mathcal{T}$. The Bagging technique [3] is used in this work in order to build a diverse pool of classifiers.

3.2 Meta-Training

In this phase, the meta-features are computed and used to train the meta-classifier λ. We select five subset of meta-features derived from the three categories presented in Section 2. As shown in Figure I-1, the meta-training stage consists of three steps: sample selection, meta-features extraction process and meta-training. A different dataset $\mathcal{T}_\lambda$ is used in this phase to prevent overfitting.

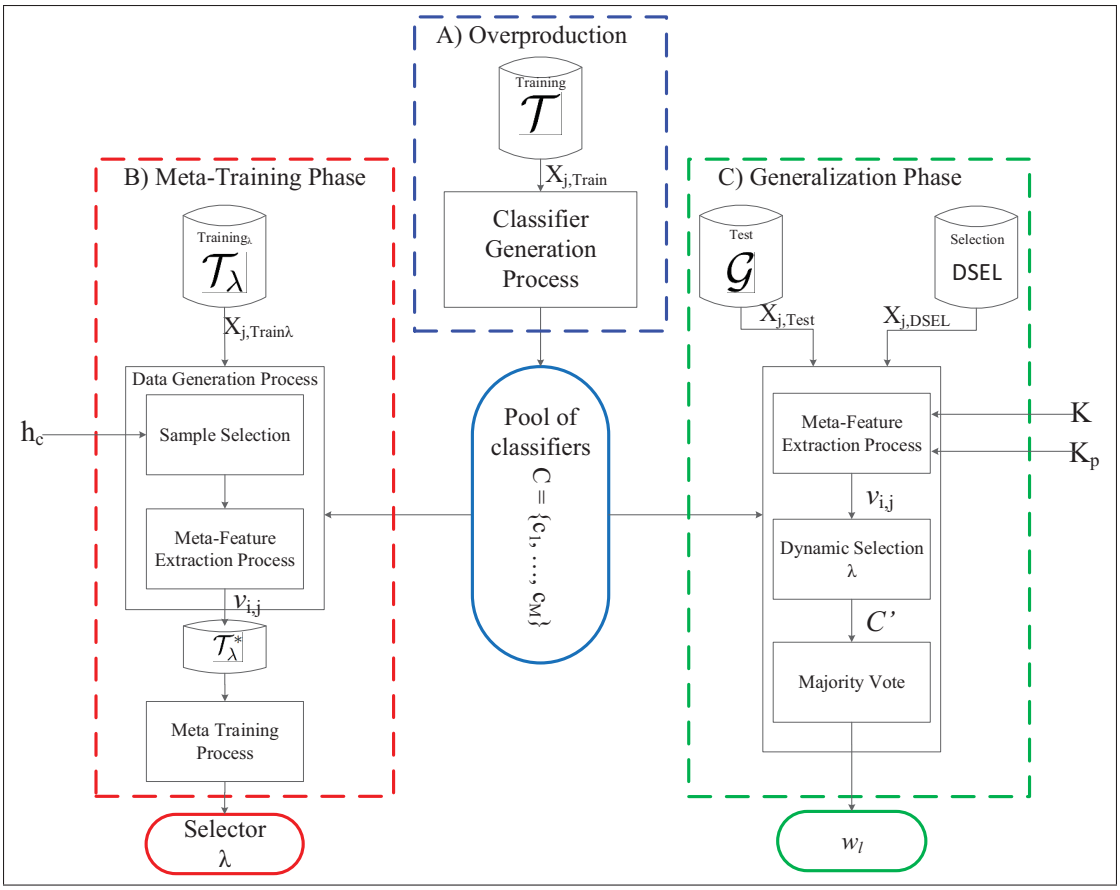


Figure-A I-1 Overview of the proposed framework. It is divided into three steps 1) Overproduction, where the pool of classifiers $C = \{c_1, \ldots, c_M\}$ is generated, 2) The training of the selector λ (meta-classifier), and 3) The generalization phase where an ensemble C' is dynamically defined based on the meta-information extracted from $\mathbf{x}_{j,test}$ and the pool $C = \{c_1, \ldots, c_M\}$. The generalization phase returns the label w_l of $\mathbf{x}_{j,test}$. h_C, K and K_p are the hyper-parameters required by the proposed system

3.2.1 Sample selection

We focus the training of λ on cases in which the extent of consensus of the pool is low. Thus, we employ a sample selection mechanism based on a threshold h_C, called the consensus threshold. For each $\mathbf{x}_{j,train_\lambda} \in \mathcal{T}_\lambda$, the degree of consensus of the pool, denoted by $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$, is computed. If $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$ falls below the threshold/ h_C, $\mathbf{x}_{j,train_\lambda}$ is passed down to the meta-features extraction process.

3.2.2 Meta-feature extraction

The first step in extracting the meta-features is to compute the region of competence of $\mathbf{x}_{j,train_\lambda}$, denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. The region of competence is defined in the $\mathcal{T}_\lambda$ set using the K-Nearest Neighbor algorithm. Then, $\mathbf{x}_j$ is transformed into an output profile, $\tilde{\mathbf{x}}_j$ by applying the transformation T (Section 2.2). The similarity between $\tilde{\mathbf{x}}_j$ and the output profiles of the instances in $\mathcal{T}_\lambda$ is obtained through the Manhattan distance. The most similar output profiles are selected to form the set $\phi_j = \left\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{K_p}\right\}$, where each output profile $\tilde{\mathbf{x}}_k$ is associated with a label $w_{l,k}$. Next, for each base classifier $c_i \in C$, five sets of meta-features are calculated:

f_1 **- Neighbors' hard classification:** First, a vector with K elements is created. For each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of the vector is set to 1, otherwise it is 0. Thus, K meta-features are computed.

f_2 **- Posterior probability:** First, a vector with K elements is created. Then, for each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, the posterior probability of c_i, $P(w_l \mid \mathbf{x}_k)$ is computed and inserted into the k-th position of the vector. Consequently, K meta-features are computed.

f_3 **- Overall Local accuracy:** The accuracy of c_i over the whole region of competence θ_j is computed and encoded as f_3.

f_4 **- Output profiles classification:** First, a vector with K_p elements is generated. Then, for each member $\tilde{\mathbf{x}}_k$ belonging to the set of output profiles ϕ_j, if the label produced by c_i for $\mathbf{x}_k$ is equal to the label $w_{l,k}$ of $\tilde{\mathbf{x}}_k$, the k-th position of the vector is set to 1, otherwise it is 0. A total of K_p meta-features are extracted using output profiles.

f_5 **- Classifier's Confidence:** The perpendicular distance between the input sample $\mathbf{x}_{j,train_\lambda}$ and the decision boundary of the base classifier c_i is calculated and encoded as f_5. f_5 is normalized to a $[0-1]$ range using the Min-max normalization.

A vector $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ is obtained at the end of the process. If c_i correctly classifies $\mathbf{x}_j$, the class attribute of $v_{i,j}$, $\alpha_{i,j} = 1$ (i.e., $v_{i,j}$ corresponds to the behavior of a competent classifier), otherwise $\alpha_{i,j} = 0$. $v_{i,j}$ is stored in the meta-features dataset $\mathcal{T}^*_\lambda$.

3.2.3 Training

The last step of the meta-training phase is the training of λ. The dataset $\mathcal{T}_{\lambda}^{*}$ is divided on the basis of 75% for training and 25% for validation. A Multi-Layer Perceptron (MLP) neural network with 10 neurons in the hidden layer is used as the meta-classifier λ. The training process is stopped if its performance on the validation set decreases or fails to improve for five consecutive epochs.

3.3 Generalization

Given an input test sample $\mathbf{x}_{j,test}$ from the generalization dataset $\mathcal{G}$, first, the region of competence θ_j and the set of output profiles ϕ_j, are calculated using the samples from the dynamic selection dataset D_{SEL}. For each classifier $c_i \in C$, the meta-features are extracted (Section 3.2.2), returning the meta-features vector $v_{i,j}$.

Next, $v_{i,j}$ is passed down as input to the meta-classifier λ, which decides whether c_i is competent enough to classify $\mathbf{x}_{j,test}$. If c_i is considered competent, it is inserted into the ensemble $C^{'}$. After each classifier of the pool is evaluated, the majority vote rule [9] is applied over the ensemble C', giving the label w_l of $\mathbf{x}_{j,test}$. Tie-breaking is handled by choosing the class with the highest a posteriori probability.

Table-A I-1 Mean and standard deviation results of the accuracy for the three scenarios. The best results are in bold. Results that are significantly better ($p < 0.05$) are underlined

Datasets	DES_D	DES_I	DES_{ALL}	λ_D	λ_I	λ_{ALL}
Pima	**77.74(2.34)**	72.14(3.69)	77.18(2.99)	**73.20 (3.48)**	68.53(1.79)	72.57(2.12)
Liver	**68.83(5.57)**	59.22(3.64)	65.53(3.20)	**68.92(2.22)**	52.90(3.66)	62.29(3.14)
Breast	**97.41(1.07)**	96.99(3.64)	96.96(1.00)	**97.54(1.04)**	85.66(6.84)	96.97(1.15)
Blood	**79.14(1.88)**	75.39(5.55)	75.79(2.62)	**82.83(5.57)**	69.32(2.90)	74.28(2.87)
Banana	**90.16(2.09)**	82.52(13.24)	85.98(1.73)	**91.14(3.09)**	83.58(6.09)	80.21(8.97)
Vehicle	82.50(2.07)	80.25(3.73)	**83.53(1.26)**	82.38(2.34)	73.70(3.85)	**88.67(3.15)**
Lithuanian	**90.26(2.78)**	79.48(13.56)	87.40(1.87)	**89.42(3.41)**	82.20(6.31)	81.70(3.97)
Sonar	79.72(1.86)	53.14(6.66)	**80.38(4.32)**	**76.15(2.43)**	60.70(7.34)	75.42(2.91)
Ionosphere	**89.31(0.95)**	86.69(6.94)	88.97(2.51)	89.18(2.31)	67.44(3.42)	**89.52(3.72)**
Wine	**96.94(3.12)**	94.39(10.91)	95.11(6.69)	**93.33(1.56)**	90.86(4.49)	78.11(6.69)
Haberman	76.71(3.52)	72.77(6.34)	**77.63(2.55)**	**76.31(2.35)**	71.88(2.72)	76.23(4.91)

4. Experiments

We evaluated the generalization performance of the proposed technique using eleven classification datasets, nine from the UCI machine learning repository, and two, artificially generated using the Matlab PRTOOLS toolbox[2]. The experiment was conducted using 20 replications. For each replication, the datasets were randomly divided on the basis of 25% for training ($\mathcal{T}$), 25% for meta-training $\mathcal{T}_\lambda$, 25% for the dynamic selection dataset (D_{SEL}) and 25% for generalization ($\mathcal{G}$). The divisions were performed maintaining the prior probability of each class. The pool of classifiers was composed of 10 Perceptrons. The value of the hyper-parameters K, K_p and h_c were 7, 5 and 70% respectively. They were selected empirically based on previous results [20].

We evaluate three different scenarios for the training of the meta-classifier λ. For the following definitions, let $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{11}\}$ be the eleven classification problems considered in this paper, and $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{11}\}$ a set of meta-classifiers trained using the meta-training dataset, $\mathcal{T}^*_{\lambda,i}$ related to a classification problem $\mathcal{D}_i$.

a. *Scenario I - λ dependent(λ_D):* The selector λ_i is trained using the meta-training data $\mathcal{T}^*_{\lambda,i}$, and is used as the classifier selector for the same classification problem $\mathcal{D}_i$. This scenario is performed in order to answer the first research question of this paper: Can the use of meta-features lead to a more robust dynamic selection technique?

b. *Scenario II - λ independent(λ_I):* The selector λ_i is trained using the meta-training data $\mathcal{T}^*_{\lambda,i}$, and is used as the classifier selector for a different classification problem $\mathcal{D}_j \mid i \neq j$. The objective of this scenario is to answer the second question posed in this work: Is the training of the meta-classifier application independent?

c. *Scenario III - λ_{ALL}:* Here, we train a single meta-classifier λ_{ALL} using the meta-training data derived from all classification problems $\mathcal{D}_i \in \mathcal{D}$, $\mathcal{T}^*_{\lambda,ALL} = \left\{\mathcal{T}^*_{\lambda,1} \cup \mathcal{T}^*_{\lambda,2} \cup \ldots, \cup \mathcal{T}^*_{\lambda,11}\right\}$. The objective of this scenario is to answer the third question posed in this paper: Can we

[2] www.prtools.org

improve the performance of the meta-classifier using knowledge from different classification problems?

For the rest of this paper, we refer to each scenario as λ_D, λ_I and λ_{ALL}. We refer to DES_D, DES_I and DES_{ALL}, the DES system created using each training scenario, respectively.

4.1 Results

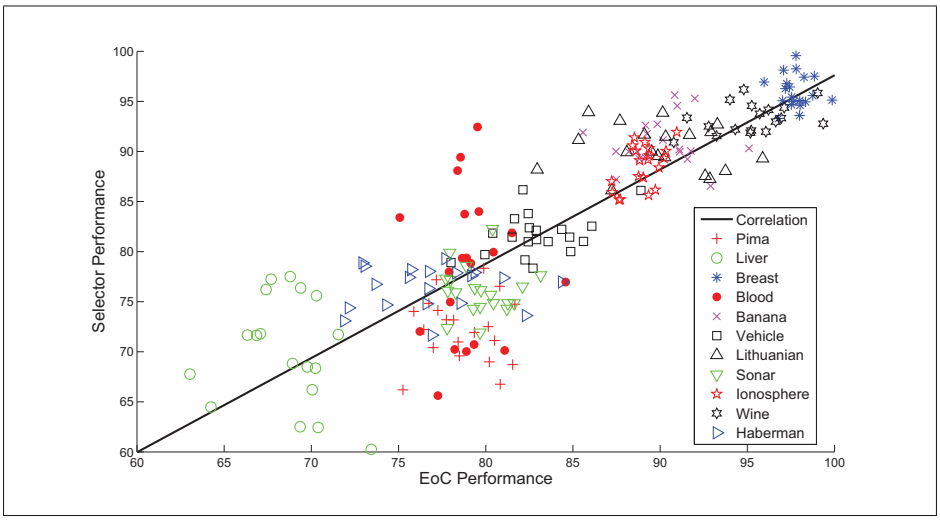


Figure-A I-2 Correlation between the performances of the proposed DES_D and λ_D. $\rho = 0.88$

Table I-1 shows a comparison of the results achieved related to scenarios I, II and III. Both the DES performance and the meta-classifier performance are presented. We compare each pair of results using the Kruskal-Wallis non-parametric statistical test with a 95% confidence interval. Results that improved the accuracy significantly are underlined.

The λ-dependent scenario (DES_D) obtained the best results. The only exception is for the Vehicle problem, where the λ_{ALL} achieved the best result. Furthermore, when the performance of the meta-classifier is significantly better, the accuracy of the DES system is also significantly better. This finding shows how the performance of the meta-classifier is correlated with the accuracy of its corresponding DES system. The independent scenario, λ_I, presented the lowest

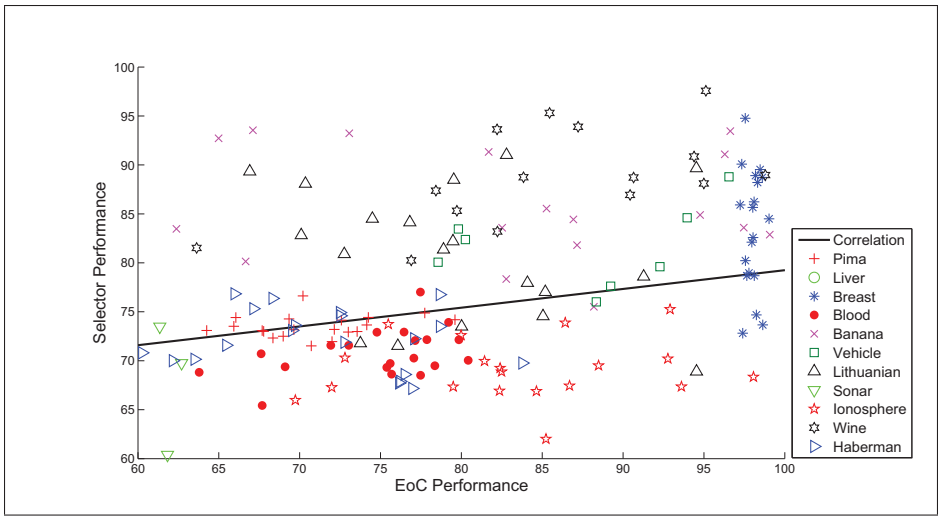


Figure-A I-3 Correlation between the performances of the proposed DES_I and λ_I. $\rho = 0.42$

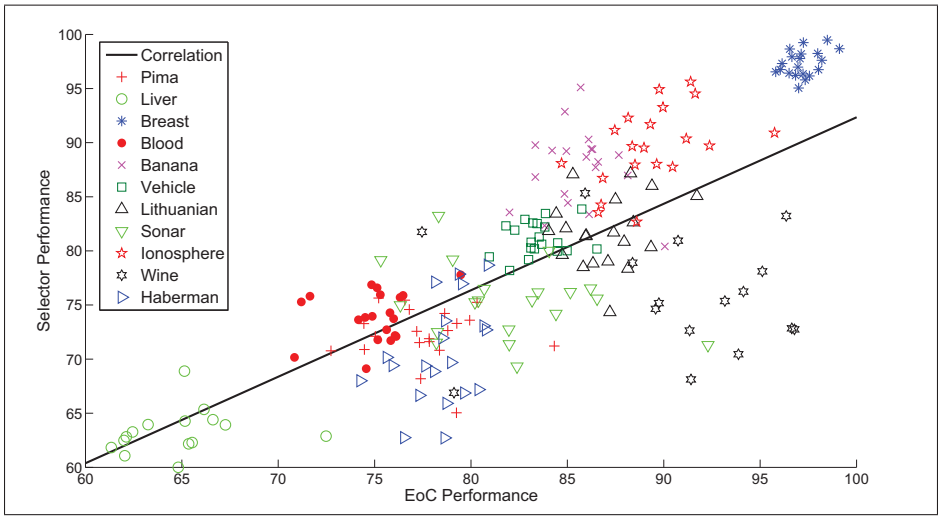


Figure-A I-4 Correlation between the performances of the proposed DES_{ALL} and λ_{ALL}. $\rho = 0.76$

results for both the DES system (DES_I) and meta-classifier (λ_I) in all cases. The accuracies of λ_I and DES_I are also significantly worse when compared to the other two scenarios.

We also study the correlation between the accuracy of the DES system and the performance of the meta-classifier for the three scenarios. Figures I-2, I-3 and I-4 show the correlation between the accuracy of the proposed DES system and the performance of the meta-classifier for the

λ_D, λ_I and λ_{ALL} scenarios, respectively. We compute the correlation coefficient, ρ, using the Pearson's Product-Moment.

Scenario I achieved the highest correlation coefficient $\rho = 0.88$, while Scenario III λ_{ALL} presented a slightly lower coefficient, $\rho = 0.76$. Thus, the use of knowledge from a different classification problem also reduced the correlation between the meta-classifier and the accuracy of the DES system. The correlation between λ_I and DES_I was $\rho = 0.42$, which is significantly lower than Scenarios I and III.

Table-A I-2 Mean and standard deviation results of the accuracy obtained for the proposed DES_D and the DES systems in the literature. The best results are in bold. Results that are significantly better ($p < 0.05$) are underlined

Database	DES_D	KNORA-E	KNORA-U	DES-FA	LCA	OLA	KNOP
Pima	**77.74(2.34)**	73.16(1.86)	74.62(2.18)	76.04(1.61)	72.86(2.98)	73.14(2.56)	73.42(2.11)
Liver Disorders	**68.92(2.22)**	63.86(3.28)	64.41(3.76)	65.72(3.81)	62.24(4.01)	62.05(3.27)	65.23(2.29)
Breast (WDBC)	**97.54(1.04)**	96.93(1.10)	96.35(1.02)	97.18(1.13)	97.15(1.58)	96.85(1.32)	95.42(0.89)
Blood Transfusion	**79.14(1.88)**	74.59(2.62)	75.50(2.36)	76.42(1.16)	72.20(2.87)	72.33(2.36)	77.54(2.03)
Banana	**90.16(2.09)**	88.83(1.67)	89.03(2.87)	**90.16(3.18)**	89.28(1.89)	89.40(2.15)	85.73(10.65)
Vehicle	**82.5(2.07)**	81.19(1.54)	82.08(1.70)	80.20(4.05)	80.33(1.84)	81.50(3.24)	80.09(1.47)
Lithuanian Classes	90.26(2.78)	88.83(2.50)	87.95(2.64)	**92.23(2.46)**	88.10(2.20)	87.95(1.85)	89.33(2.29)
Sonar	**79.72(1.86)**	74.95(2.79)	76.69(1.94)	77.52(1.86)	76.51(2.06)	74.52(1.54)	75.72(2.82)
Ionosphere	**89.31(0.95)**	87.37(3.07)	86.22(1.67)	86.33(2.12)	86.56(1.98)	86.56(1.98)	85.71(5.52)
Wine	**96.94(3.12)**	95.00(1.53)	96.13(1.62)	95.45(1.77)	95.85(2.25)	96.16(3.02)	95.00(4.14)
Haberman	**76.71(3.52)**	71.23(4.16)	74.40(2.27)	74.47(2.41)	70.16(3.56)	72.26(4.17)	75.00(3.40)

Therefore, experimental results indicate that the training of the meta-classifier is problem-dependent. The behavior of a competent classifier differs according to each classification problem. Furthermore, as the λ_{ALL} selector performed worse than the λ_D, we failed to improve the performance of the meta-classifier and DES system by adding knowledge derived from other classification problems. However, the loss in accuracy might be explained by the use of classification problems with completely different distributions and data complexities [105].

4.2 Comparison with the state-of-the-art

In Table I-2, we compare the recognition rates obtained by the proposed DES_D against dynamic selection techniques in the literature (KNORA-Eliminate [14], KNORA-Union [14], DES-

FA [20], LCA [22], OLA [22] and KNOP [16]). We compare each pair of results using the Kruskal-Wallis non-parametric statistical test with a 95% confidence interval. The results of the proposed DES_D over the Pima, Liver Disorders, Blood Transfusion, Vehicle, Sonar and Ionosphere datasets are statistically superior to the result of the best DES from the literature. For the other datasets, Breast, Banana and Lithuanian, the results are statistically equivalent.

We can thus answer the first question posed in this paper: Can the use of meta-features lead to a more robust dynamic selection technique? As the result of the proposed DES_D is significantly better in eight datasets, the use of meta-learning indeed leads to a more robust dynamic ensemble selection technique.

5. Conclusion

In this paper, we present a novel DES framework using meta-learning. Different properties of the behavior of a base classifier are extracted from the training data and encoded as meta-features. These meta-features are used to train a meta-classifier that can estimate whether a base classifier is competent enough to classify a given input sample. Based on the proposed framework, we perform three experiments considering three different scenarios for the training of the meta-classifier.

Experimental results show that the training of the proposed meta-classifier is problem-dependent as the dependent scenario, λ_D, outperformed both λ_I and λ_{ALL}. In addition, the correlation between the performances of λ_D and the accuracy of the corresponding DES_D is also higher than that of the other two scenarios.

A comparison with the state-of-the-art dynamic ensemble selection techniques shows that the proposed DES_D outperforms current techniques. Moreover, the gain in accuracy observed with our system is also statistically significant. Thus, we can conclude that the use of multiple properties of the behavior of a base classifier in the classification environment indeed leads to a more robust DES system.

Future works on this topic will involve:

a. The evaluation of a different training scenario using only classification problems with similar data complexity for the training of the meta-classifier.

b. the design of new meta-features in order to improve the performance of the meta-classifier, and consequently, the DES system.

APPENDIX II

META-DES.H: A DYNAMIC ENSEMBLE SELECTION TECHNIQUE USING META-LEARNING AND A DYNAMIC WEIGHTING APPROACH

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Published in « International Joint Conference on Neural Network (IJCNN) » 2015.

Abstract

In Dynamic Ensemble Selection (DES) techniques, only the most competent classifiers are selected to classify a given query sample. Hence, the key issue in DES is how to estimate the competence of each classifier in a pool to select the most competent ones. In order to deal with this issue, we proposed a novel dynamic ensemble selection framework using meta-learning, called META-DES. The framework is divided into three steps. In the first step, the pool of classifiers is generated from the training data. In the second phase the meta-features are computed using the training data and used to train a meta-classifier that is able to predict whether or not a base classifier from the pool is competent enough to classify an input instance. In this paper, we propose improvements to the training and generalization phase of the META-DES framework. In the training phase, we evaluate four different algorithms for the training of the meta-classifier. For the generalization phase, three combination approaches are evaluated: Dynamic selection, where only the classifiers that attain a certain competence level are selected; Dynamic weighting, where the meta-classifier estimates the competence of each classifier in the pool, and the outputs of all classifiers in the pool are weighted based on their level of competence; and a hybrid approach, in which first an ensemble with the most competent classifiers is selected, after which the weights of the selected classifiers are estimated in order to be used in a weighted majority voting scheme. Experiments are carried out on 30 classification datasets.

Experimental results demonstrate that the changes proposed in this paper significantly improve the recognition accuracy of the system in several datasets.

1. Introduction

Multiple Classifier Systems (MCS) aim to combine classifiers in order to increase the recognition accuracy in pattern recognition systems [24; 9]. MCS are composed of three phases [1]: (1) Generation, (2) Selection, and (3) Integration. In the first phase, a pool of classifiers is generated. In the second phase, a single classifier or a subset having the best classifiers of the pool is(are) selected. We refer to the subset of classifiers as the Ensemble of Classifiers (EoC). In the last phase, integration, the predictions of the selected classifiers are combined to obtain the final decision [24].

Recent works in MCS have shown that dynamic ensemble selection (DES) techniques achieve higher classification accuracy when compared to static ones [1; 2; 14]. This is specially true for ill-defined problems, i.e., for problems where the size of the training data is small and there are not enough data available to train the classifiers [16; 17]. The key issue in DES is to define a criterion to measure the level of competence of a base classifier. Most DES techniques [14; 22; 21; 20] use estimates of the classifiers' local accuracy in small regions of the feature space surrounding the query instance, called the region of competence, as a search criterion to estimate the competence level of the base classifier. However, in our previous work [20], we demonstrated that the use of local accuracy estimates alone is insufficient to provide higher classification performance.

To tackle this issue, in [2] we proposed a novel DES framework, called META-DES, in which multiple criteria regarding the behavior of a base classifier are used to compute its level of competence. The framework is based on two environments: the classification environment, in which the input features are mapped into a set of class labels, and the meta-classification environment, where different properties from the classification environment, such as the classifier accuracy in a local region of the feature space, are extracted from the training data and

encoded as meta-features. With the arrival of new test data, the meta-features are extracted using the test data as reference, and used as input to the meta-classifier. The meta-classifier decides whether the base classifier is competent enough to classify the test sample. The framework is divided into three steps: (1) Overproduction, where the pool of classifiers is generated; (2) Meta-training, where the meta-features are extracted, using the training data, and used as inputs to train a meta-classifier that works as a classifier selector, and (3) the Generalization phase, in which the meta-features are extracted from each query sample and used as input to the meta-classifier to perform the ensemble selection.

In this paper, we propose two improvements to the META-DES framework. First, we modify the training routine of the meta-classifier. The modification made is motivated by the fact that there is a strong correlation between the performance of the meta-classifier for the selection of "competent" classifiers, i.e., classifiers that predict the correct label for a given query sample and the classification accuracy of the DES system [36]. Hence, we believe that the proposed META-DES framework can obtain higher classification performance by focusing only on improving the performance of the system at the meta-classification level. This is an interesting feature of the proposed system especially when dealing with ill-defined problems due to critical dataset sizes [2]. Four different classifier models are considered for the meta-classifier: MLP Neural Network, Support Vector Machines with Gaussian Kernel (SVM), s and Naive Bayes [71].

Secondly, we propose three combination schemes for the generalization phase of the framework: Dynamic selection, Dynamic weighting and Hybrid. In the dynamic selection approach, only the classifiers that attain a certain level of competence are used to classify a given query sample. In the dynamic weighting approach, the meta-classifier is used to estimate the weights of all base classifiers in the pool. Then, their decisions are aggregated using a weighted majority voting scheme [9]. Thus, classifiers that attain a higher level of competence, for the classification of the given query sample, have a greater impact on the final decision. In the hybrid approach, only the classifiers that attain a certain level of competence are selected. Then, the meta-classifier is used to compute the weights of the selected base classifiers to be used in

a weighted majority voting scheme. The hybrid approach is based on the observation that the selected base classifiers might be associated with different levels of competence. It is feasible that classifiers that attained a higher level of competence should have more influence for the classification of the given test sample. The proposed framework differs from mixture of expert techniques [106; 26], since our system is based on the mechanism used for the selection of dynamic ensembles [1; 2] rather than static ones [26]. In addition, mixture of experts techniques are dedicated to the use of neural networks as base classifier, while, in the proposed framework, any classification algorithm can be used.

We evaluate the generalization performance of the system over 30 classification problems derived from different data repositories. Furthermore, the recognition performance of the system is compared against eight state-of-the-art dynamic selection techniques according to a new survey on this topic [1]. Experimental results demonstrate that the choice of the meta-classifier has a significant impact on the classification accuracy of the overall system. The modifications proposed in this work significantly improve the performance of the framework when compared to state-of-the-art dynamic selection techniques.

This paper is organized as follows: The META-DES framework is introduced in Section 2. Experimental results are given in Section 3. Finally the conclusion is presented in the last section.

2. The META-DES Framework

The META-DES framework is based on the assumption that the dynamic ensemble selection problem can be considered as a meta-problem. This meta-problem uses different criteria regarding the behavior of a base classifier c_i, in order to decide whether it is competent enough to classify a given test sample $\mathbf{x}_j$. The meta-problem is defined as follows [2]:

- The **meta-classes** of this meta-problem are either “competent” (1) or “incompetent” (0) to classify $\mathbf{x}_j$.

- Each set of **meta-features** f_i corresponds to a different criterion for measuring the level of competence of a base classifier.
- The meta-features are encoded into a **meta-features vector** $v_{i,j}$.
- A **meta-classifier** λ is trained based on the meta-features $v_{i,j}$ to predict whether or not c_i will achieve the correct prediction for $\mathbf{x}_j$, i.e., if it is competent enough to classify $\mathbf{x}_j$

A general overview of the META-DES framework is depicted in Figure II-1. It is divided into three phases: Overproduction, Meta-training and Generalization.

2.1 Overproduction

In this step, the pool of classifiers $C = \{c_1, \ldots, c_M\}$, where M is the pool size, is generated using the training dataset $\mathcal{T}$. The Bagging technique [3] is used in this work in order to build a diverse pool of classifiers.

2.2 Meta-Training

In this phase, the meta-features are computed and used to train the meta-classifier λ. As shown in Figure II-1, the meta-training stage consists of three steps: sample selection, meta-features extraction process and meta-training. A different dataset $\mathcal{T}_\lambda$ is used in this phase to prevent overfitting.

2.2.1 Sample selection

We decided to focus the training of λ on cases in which the extent of consensus of the pool is low. This decision was based on the observations made in [15; 16] the main issues in dynamic ensemble selection occur when classifying testing instances where the degree of consensus among the pool of classifiers is low, i.e., when the number of votes from the winning class is close to or even equal to the number of votes from the second class. We employ a sample selection mechanism based on a threshold h_C, called the consensus threshold. For each

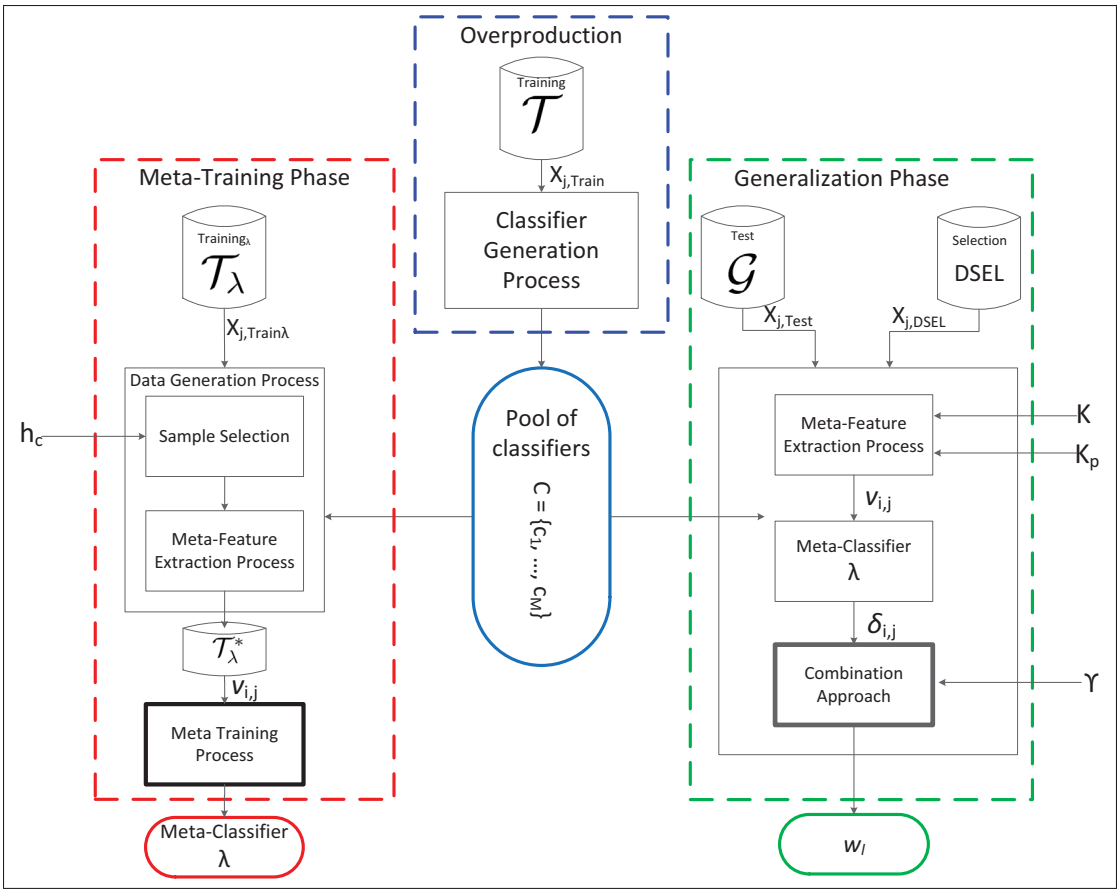


Figure-A II-1 Overview of the proposed framework. It is divided into three steps 1) Overproduction, where the pool of classifiers $C = \{c_1, \ldots, c_M\}$ is generated, 2) The training of the selector λ (meta-classifier), and 3) The generalization phase where the level of competence $\delta_{i,j}$ of each base classifier c_i is calculated specifically for each new test sample $\mathbf{x}_{j,test}$. Then, the level of competence $\delta_{i,j}$ is used by the combination approach to predict the label w_l of the test sample $\mathbf{x}_{j,test}$. Three combination approaches are considered: Dynamic selection (META-DES.S), Dynamic weighting (META-DES.W) and Hybrid (META-DES.H). h_C, K, K_p and Υ are the hyper-parameters required by the proposed system [Adapted from [2]]

$\mathbf{x}_{j,train_\lambda} \in \mathcal{T}_\lambda$, the degree of consensus of the pool, denoted by $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$, is computed. If $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$ falls below the threshold h_C, $\mathbf{x}_{j,train_\lambda}$ is passed down to the meta-features extraction process.

2.2.2 Meta-feature extraction

The first step in extracting the meta-features involves computing the region of competence of $\mathbf{x}_{j,train_\lambda}$, denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. The region of competence is defined in the $\mathcal{T}_\lambda$ set using the K-Nearest Neighbor algorithm. Then, $\mathbf{x}_{j,train_\lambda}$ is transformed into an output profile, $\tilde{\mathbf{x}}_{j,train_\lambda} = \left\{\tilde{\mathbf{x}}_{j,train_\lambda,1}, \tilde{\mathbf{x}}_{j,train_\lambda,2}, \ldots, \tilde{\mathbf{x}}_{j,train_\lambda,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,train_\lambda,i}$ is the decision yielded by the base classifier c_i for the sample $\mathbf{x}_{j,train_\lambda}$ [16].

The similarity between $\tilde{\mathbf{x}}_{j,train_\lambda}$ and the output profiles of the instances in $\mathcal{T}_\lambda$ is obtained through the Euclidean distance. The most similar output profiles are selected to form the set $\phi_j = \left\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{K_p}\right\}$, where each output profile $\tilde{\mathbf{x}}_k$ is associated with a label $w_{l,k}$. Next, for each base classifier $c_i \in C$, five sets of meta-features are calculated:

f_1 **- Neighbors' hard classification:** First, a vector with K elements is created. For each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of the vector is set to 1, otherwise it is 0. Thus, K meta-features are computed.

f_2 **- Posterior probability:** First, a vector with K elements is created. Then, for each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, the posterior probability of c_i, $P(w_l \mid \mathbf{x}_k)$ is computed and inserted into the k-th position of the vector. Consequently, K meta-features are computed.

f_3 **- Overall Local accuracy:** The accuracy of c_i over the whole region of competence θ_j is computed and encoded as f_3.

f_4 **- Output profiles classification:** First, a vector with K_p elements is generated. Then, for each member $\tilde{\mathbf{x}}_k$ belonging to the set of output profiles ϕ_j, if the label produced by c_i for $\mathbf{x}_k$ is equal to the label $w_{l,k}$ of $\tilde{\mathbf{x}}_k$, the k-th position of the vector is set to 1, otherwise it is 0. A total of K_p meta-features are extracted using output profiles.

f_5 **- Classifier's Confidence:** The perpendicular distance between the input sample $\mathbf{x}_{j,train_\lambda}$ and the decision boundary of the base classifier c_i is calculated and encoded as f_5. f_5 is normalized to a $[0-1]$ range using the Min-max normalization.

A vector $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ is obtained at the end of the process. If c_i correctly classifies $\mathbf{x}_j$, the class attribute of $v_{i,j}$, $\alpha_{i,j} = 1$ (i.e., $v_{i,j}$ belongs to the meta-class "competent"), otherwise $\alpha_{i,j} = 0$. $v_{i,j}$ is stored in the meta-features dataset $\mathcal{T}_\lambda^*$ that is used to train the meta-classifier λ.

2.2.3 Training

The last step of the meta-training phase is the training of λ. The dataset $\mathcal{T}_\lambda^*$ is divided on the basis of 75% for training and 25% for validation. In this paper, we evaluate four classifier models for the meta-classifier: MLP Neural Network, Support Vector Machines with Gaussian Kernel (SVM), Random Forests and Naive Bayes. These classifiers were selected based on a recent study [71] that ranked the best classification models in a comparison considering a total of 179 classifiers and 121 datasets. All classifiers were implemented using the Matlab PRTOOLS toolbox [63]. The parameters of each classifier were set as follows:

a. MLP Neural Network: The validation data was used to select the number of nodes in the hidden layer. We used a configuration with 10 neurons in the hidden layer since there were no improvement in results with more than 10 neurons. The training process for λ was performed using the Levenberg-Marquadt algorithm. The training process was stopped if its performance on the validation set decreased or failed to improve for five consecutive epochs.

b. SVM: A radial basis SVM with a Gaussian Kernel was used. For each dataset, a grid search was performed in order to set the values of the regularization parameter c and the Kernel spread parameter γ.

c. Random Forest: A total of 200 decision trees were used. The depth of each tree was fixed at 5.

d. Naive Bayes: A simple Naive Bayes classifier using a normal distribution to model numeric features. No parameters are required for this model.

2.3 Generalization

Given the query sample $\mathbf{x}_{j,test}$, the region of competence θ_j is computed using the samples from the dynamic selection dataset D_{SEL}. Following that, the output profiles $\tilde{\mathbf{x}}_{j,test}$ of the test sample, $\mathbf{x}_{j,test}$, are calculated. The set with K_p similar output profiles ϕ_j, of the query sample $\mathbf{x}_{j,test}$, is obtained through the Euclidean distance applied over the output profiles of the dynamic selection dataset, $\tilde{D}_{SEL}$.

Next, for each classifier c_i belonging to the pool of classifiers C, the meta-features extraction process is called, returning the meta-features vector $v_{i,j}$. Then, $v_{i,j}$ is used as input to the meta-classifier λ. The support obtained by the meta-classifier for the "competent" meta-class, denoted by $\delta_{i,j}$, is computed as the level of competence of the base classifier c_i for the classification of the test sample $\mathbf{x}_{j,test}$.

Three combination approaches are considered:

- **META-DES.S**: In this approach, the base classifiers that achieve a level of competence $\delta_{i,j} > \Upsilon$ are considered competent, and are selected to compose the ensemble C'. In this paper, we set $\Upsilon = 0.5$ (i.e., the base classifier is selected if the support for the "'competent"' meta-class is higher than the support for the "'incompetent"' meta-class). The final decision is obtained using the majority vote rule [24]. Tie-breaking is handled by choosing the class with the highest a posteriori probability.

- **META-DES.W**: Every classifier in the pool C is used to predict the label of $\mathbf{x}_{j,test}$. The level of competence $\delta_{i,j}$ estimated by the meta-classifier λ is used as the weight of each base classifier. The final decision is obtained using a weighted majority vote combination scheme [9]. Thus, the decisions obtained by the base classifiers with a higher level of competence $\delta_{i,j}$ have a greater influence on the final decision.

- **META-DES.H**: In this approach, first the base classifiers that achieve a level of competence $\delta_{i,j} > \Upsilon = 0.5$ are considered competent and are selected to compose the ensemble C'. Next, the level of competence $\delta_{i,j}$ estimated by the meta-classifier λ, for the classifiers in

the ensemble C', are used as its weights. Thus, the decisions obtained by the base classifiers with the highest level of competence $\delta_{i,j}$ have a greater influence in the final decision. A weighting majority voting scheme is used to predict the label w_l of $\mathbf{x}_{j,test}$.

3. Experiments

3.1 Datasets

A total of 30 datasets are used in the comparative experiments, with sixteen taken from the UCI machine learning repository [59], four from the STATLOG project [60], four from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [61], four from the Ludmila Kuncheva Collection of real medical data [62], and two artificial datasets generated with the Matlab PRTOOLS toolbox [63]. The key features of each dataset are shown in Table II-1.

Table-A II-1 Key Features of the datasets used in the experiments

Database	No. of Instances	Dimensionality	No. of Classes	Source
Pima	768	8	2	UCI
Liver Disorders	345	6	2	UCI
Breast (WDBC)	568	30	2	UCI
Blood transfusion	748	4	2	UCI
Banana	1000	2	2	PRTOOLS
Vehicle	846	18	4	STATLOG
Lithuanian	1000	2	2	PRTOOLS
Sonar	208	60	2	UCI
Ionosphere	315	34	2	UCI
Wine	178	13	3	UCI
Haberman's Survival	306	3	2	UCI
Cardiotocography (CTG)	2126	21	3	UCI
Vertebral Column	310	6	2	UCI
Steel Plate Faults	1941	27	7	UCI
WDG V1	50000	21	3	UCI
Ecoli	336	7	8	UCI
Glass	214	9	6	UCI
ILPD	214	9	6	UCI
Adult	48842	14	2	UCI
Weaning	302	17	2	LKC
Laryngeal1	213	16	2	LKC
Laryngeal3	353	16	3	LKC
Thyroid	215	5	3	LKC
German credit	1000	20	2	STATLOG
Heart	270	13	2	STATLOG
Satimage	6435	19	7	STATLOG
Phoneme	5404	6	2	ELENA
Monk2	4322	6	2	KEEL
Mammographic	961	5	2	KEEL
MAGIC Gamma Telescope	19020	10	2	KEEL

Table-A II-2 Comparison of different classifier types used as the meta-classifier λ for the META-DES framework. The best results are in bold. Results that are significantly better are marked with a •

	Meta-Classifier λ				META-DES			
Dataset	**λ MLP NN**	**λ SVM**	**λ Forest**	**λ Bayes**	**MLP NN**	**SVM**	**Forest**	**Bayes**
Pima	78.53(1.24)	79.46(1.67)	**80.27(2.08)**	79.63(1.75)	**79.03(2.24)**	77.58(1.67)	78.39(2.08)	77.76(1.75)
Liver	68.83 (5.57)	70.60(5.52)	69.56(5.17)	**71.24(4.84)**	**70.08(3.49)**	68.92(5.52)	67.88(5.17)	69.56(4.84)
Breast	95.43 (1.85)	97.19(0.61)	97.19(0.61)	**97.66(0.50)**	**97.41(1.07)**	96.94(0.61)	96.94(0.61)	96.94(0.61)
Blood	79.54(3.03)	79.18(1.88)	**79.83(2.42)**	79.66(1.52)	**79.14(1.03)**	77.84(1.88)	78.49(2.42)	78.31(1.52)
Banana	91.14(3.09)	95.17(1.75)	90.97(3.89)	**95.67(2.37)** •	91.78(2.68)	93.92(1.75)	89.72(3.89)	**94.42(2.37)** •
Vehicle	82.38(2.34)	82.50(1.92)	82.44(1.63)	**82.76(2.01)**	82.75(1.70)	83.29(1.92)	83.24(1.63)	**83.55(2.01)**
Lithuanian	93.42(3.41)	94.91(1.25)	**97.89(0.81)** •	93.72(3.09)	93.18(1.32)	94.30(1.25)	**97.28(0.81)** •	93.12(3.09)
Sonar	86.15(2.43)	85.88(4.08)	84.60(4.61)	**86.95(5.67)** •	80.55(5.39)	80.77(4.08)	79.49(4.61)	**81.84(5.67)**
Ionosphere	**89.18(2.31)**	87.35(2.42)	87.09(2.48)	87.35(2.21)	**89.94(1.96)**	89.06(2.42)	88.80(2.48)	89.06(2.21)
Wine	**98.90(1.61)**	**98.90(1.61)**	**98.90(1.61)**	97.25(1.48)	99.25(1.11)	**99.27(1.61)**	99.02(1.61)	98.53(1.48)
Haberman	**76.31(2.35)**	74.81(2.50)	75.69(2.19)	75.25(2.06)	**76.71(1.86)**	75.69(2.50)	76.56(2.19)	76.13(2.06)
CTG	82.00(5.22)	88.81(1.03)	88.60(1.04)	**90.21(1.14)** •	84.62(1.08)	85.64(1.03)	85.43(1.04)	**86.04(1.14)** •
Vertebral	86.89(2.46)	87.70(2.87)	**87.85(3.54)**	86.56(2.35)	86.89(2.46)	86.76(2.87)	**86.90(3.54)**	85.62(2.35)
Faults	70.21(4.26)	74.41(1.17)	74.41(1.17)	**74.68(1.19)** •	67.21(1.20)	68.45(1.17)	68.45(1.17)	**68.72(1.19)** •
WDVG1	83.26(1.36)	85.26(0.63)	85.23(0.50)	**85.84(0.60)** •	84.56(0.36)	84.67(0.63)	84.64(0.50)	**84.84(0.36)** •
Ecoli	77.09(4.84)	**78.01(3.89)**	76.74(3.58)	77.01(3.76)	77.25(3.52)	**80.92(3.89)** •	80.66(3.58)	80.92(3.76)
GLASS	**69.18(1.49)** •	63.31(4.40)	64.84(4.44)	64.89(3.65)	**66.87(2.99)** •	65.62(4.40)	64.16(4.44)	65.21(3.65)
ILPD	69.80(4.96)	70.48(2.17)	69.95(2.32)	**71.09(2.33)** •	69.40(1.64)	69.56(2.17)	69.03(2.32)	**70.17(2.33)**
Adult	87.00(6.29)	**88.75(1.76)**	88.68(1.29)	88.62(1.84)	87.15(2.43)	**87.35(1.76)**	87.29(1.29)	87.22(1.84)
Weaning	79.55(4.44)	79.75(2.85)	79.75(2.85)	**80.33(3.71)**	79.67(3.78)	79.10(2.85)	79.10(2.85)	**79.69(3.71)**
Laryngeal1	77.81(3.51)	80.08(3.67)	**81.29(3.79)** •	79.94(5.00)	79.67(3.78)	81.97(3.67)	**82.18(3.78)** •	81.97(5.00)
Laryngeal3	72.42(3.57)	72.63(0.87)	72.76(0.81)	**73.82(0.67)**	72.65(2.17)	73.17(2.32)	74.04(2.23)	**74.42(1.26)** •
Thyroid	96.16(5.96)	97.27(2.32)	97.15(2.23)	**97.52(1.26)**	96.78(0.87)	97.18(0.87)	97.31(0.81)	**97.38(0.67)**
German	75.00(4.18)	76.18(2.82)	**77.11(1.58)** •	75.38(1.30)	75.55(1.31)	75.34(2.82)	**76.27(2.58)**	74.54(1.30)
Heart	84.38(4.63)	83.67(2.76)	82.85(3.60)	**86.99(2.30)** •	84.80(3.36)	84.97(2.76)	84.15(3.60)	**85.30(2.30)**
Segmentation	96.89(0.74)	96.78(0.60)	96.95(0.75)	**96.99(0.60)**	96.21(0.87)	96.21(0.60)	96.38(0.75)	**96.42(0.76)**
Phoneme	80.99(3.88)	86.80(3.19)	86.80(3.19)	**90.13(0.72)** •	80.35(2.58)	78.44(3.19)	78.44(3.19)	**81.77(0.72)**
Monk2	83.89(2.59)	86.40(2.82)	85.68(2.45)	**88.67(3.32)** •	83.24(2.19)	81.08(2.82)	80.36(2.45)	**83.34(3.32)**
Mammographic	78.00(5.93)	**87.30(1.82)** •	87.30(1.53)	86.34(2.54)	84.82(1.55)	**85.37(1.82)**	85.37(1.53)	84.41(2.54)
Magic Gamma Telescope	75.40(2.25)	72.30(3.33)	74.57(3.56)	**78.65(2.52)**	84.35(3.27)	81.35(4.21)	84.35(3.27)	**85.33(2.29)**
Wilcoxon Signed Test	n/a	~ ($\rho = 0.110$)	+ ($\rho = 0.004$)	+ ($\rho = 0.007$)	n/a	~ ($\rho = 0.70$)	~ ($\rho = 0.500$)	~ ($\rho = 0.30$)

3.2 Experimental Protocol

For the sake of simplicity, the same experimental protocol used in previous publications [2; 36] was used. The experiments were carried out using 20 replications. For each replication, the datasets were randomly divided on the basis of 50% for training, 25% for the dynamic selection dataset (D_{SEL}), and 25% for the test set ($\mathcal{G}$). The divisions were performed while maintaining the prior probabilities of each class. For the proposed META-DES, 50% of the training data was used in the meta-training process $\mathcal{T}_\lambda$ and 50% for the generation of the pool of classifiers ($\mathcal{T}$).

For the two-class classification problems, the pool of classifiers was composed of 100 Perceptrons generated using the bagging technique [3]. For the multi-class problems, the pool of classifiers was composed of 100 multi-class Perceptrons. The use of Perceptron as base clas-

sifier is based on the observations that the use of weak classifiers can show more differences between the DES schemes [14], thus making it a better option for comparing different DES techniques. Furthermore, as reported by Leo Breiman, the bagging technique achieves better results when weak and unstable base classifiers are used [3].

The values of the hyper-parameters K, K_p and h_c were set at 7, 5 and 70%, respectively. They were selected empirically based on previous publications [20; 36; 2].

3.3 Comparison of different classification models as the Meta-Classifier

In this experiment, we analyze the impact of the classifier model used for the meta-problem (i.e., for the selection of competent classifiers). The objective of this experiment is to verify whether we can improve the classification performance of the META-DES system, previously defined using an MLP neural network as the meta-classifier. The following classifier models are considered: Multi-Layer Perceptron (MLP) Neural Networks as in [2], Support Vector Machines with Gaussian Kernel (SVM), Random Forests and Naive Bayes.

Table II-2 shows a comparison of the performance of the meta-classifier λ and the recognition accuracy obtained by the META-DES system using each classification model. The best results are highlighted in bold. For each dataset, we compared the results obtained by the meta-classifier λ and by the META-DES framework using the MLP network [2], against the best result obtained by any of the other classifier models (SVM, Random Forest and Naive Bayes). The comparison was performed using the Kruskal-Wallis non-parametric statistical test, with a 95% confidence interval. Results that are significantly better are marked with a $\bullet$.

We can observe that when the meta-classifier achieves a recognition performance that is statistically superior for a single dataset, such as, Banana, Faults and WDGV1, for instance, the META-DES is also likely to achieve superior accuracy for the same classification problem. Figure II-2 shows the number of datasets that each classifier model achieved the highest accuracy. The Naive Bayes classifier is ranked first, achieving the best results for 14 datasets, followed by the MLP Neural Network with 8. SVM and Random Forests achieved the best

results for 4 datasets each. The strong performance of the Naive Bayes may be explained by the fact that the majority of the meta-features are binary, and this classifier model handles well binary input features different than MLP Networks. In addition, it might indicate that the proposed sets of meta-features are possibly independent [55]. This is an interesting finding since the Naive Bayes model is much faster both in the training and testing stages when compared to an MLP Neural Network or an SVM classifier.

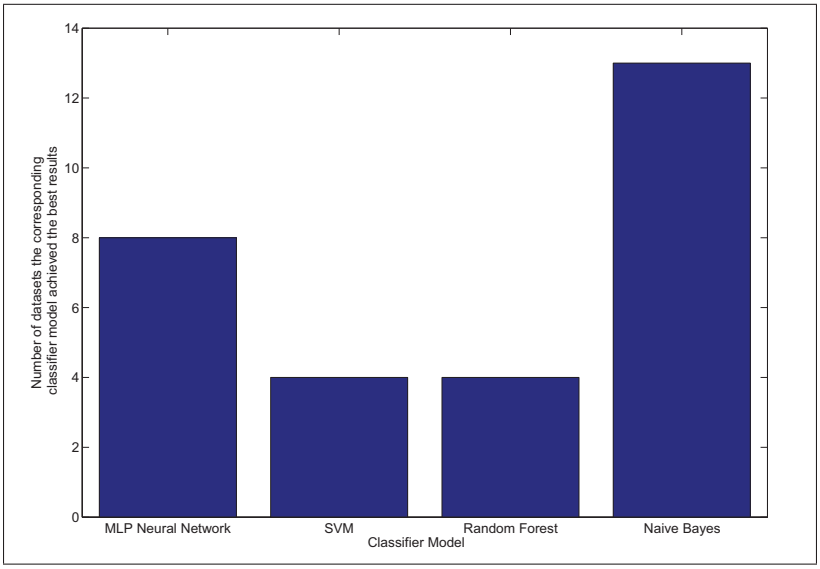


Figure-A II-2 Bar plot showing the number of datasets that each classification model used a the meta-classifier λ presented the highest recognition accuracy

Furthermore, in order to verify whether the difference in classification results obtained over the 30 datasets is statistically significant, we performed a Wilcoxon non-parametric signed rank test with 95% confidence for a pairwise comparison between the results obtained using an MLP Neural Network against the best result obtained using a different classifier for the meta-classifier. The Wilcoxon signed rank test was used since it was suggested in [91] as a robust method for comparing the classification results of two algorithms over several datasets. The results of the Wilcoxon statistical test are shown in the last row of Table II-2. Techniques that achieve performance equivalent to the MLP network are marked with "~"; those that achieve statistically superior performance are marked with a "+", and those with inferior performance

are marked with a "-". When comparing the performance of the four meta-classifiers, the results achieved using Random Forests and Naive Bayes as the meta-classifier λ are significantly superior.

Hence, we can conclude that significant gains in classification accuracy can be achieved by choosing a more suitable classifier model for the meta-classifier λ. Although the choice of the best meta-classifier may vary according to the classification problem (Table II-2), the results of the META-DES using Naive Bayes as the meta-classifier achieves results that are statistically superior when compared to the MLP neural network over the 30 datasets studied in this work.

3.4 Comparison Between Combination Approaches: Dynamic Selection, Dynamic Weighting and Hybrid

In this section, we compare the three combination approaches presented in Section 2.3: Dynamic Selection, Dynamic weighting, and the Hybrid approach. For the sake of simplicity, we present only the results obtained using the Naive Bayes as the meta-classifier λ since it achieved the highest classification accuracy in the previous experiments (Table II-2).

The results achieved using the Naive Bayes as meta-classifier for the three combination approaches are shown in Table II-3. In order to select the best combination approach, we compare the average ranks of each approach computed using the Friedman test, which is a non-parametric equivalent of the repeated measures ANOVA used to compare several algorithms over multiple datasets [107; 91]. The Friedman test ranks each algorithm, with the best performing one getting rank 1, the second best rank 2, and so forth for each dataset separately. The average rank is then computed, considering all datasets. Thus, the best algorithm is the one with the lowest average rank. The approaches that use the proposed weighting scheme (Dynamic weighting and Hybrid) outperformed the Dynamic selection approach in accuracy. This can be explained by the fact the outputs given by the Naive Bayes classifier can be directly interpreted as the likelihood that the base classifier belongs to the "‘competent"’ meta-class. Thus, the supports provided by the meta-classifier can directly be used as the weights of each

classifier for a weighted majority voting scheme. This is different from other classification models, such as Random Forests where their class supports cannot be directly interpreted as such. Hence, the meta-classifier can also be used for the fusion (integration) of the classifiers in the ensemble, rather than only for ensemble selection. Since the Hybrid combination approach presents the highest recognition accuracy when the 30 datasets are considered (lowest average rank) this combination approach is selected for the comparison against other state-of-the-art DES techniques.

Table-A II-3 Comparison between the three classification approaches: Selection, Weighting and Hybrid for the META-DES framework. The results using a Naive Bayes as the meta-classifier λ are presented. The best results are in bold. The average rank is shown in the last row of the table

Dataset	META-DES.S	META-DES.W	META-DES.H
Pima	77.76(1.75)	77.64(1.68)	**77.93(1.86)**
Liver	69.56(4.84)	69.69(4.68)	**69.95(3.49)**
Breast	**97.41(0.50)**	97.25(0.47)	97.25(0.47)
Blood	78.31(1.52)	**78.67(1.77)**	78.25(1.37)
Banana	94.42(2.37)	**95.13(1.88)**	94.51(2.36)
Vehicle	**83.55(2.01)**	83.50(1.87)	83.55(2.10)
Lithuanian	93.12(3.09)	93.19(3.14)	**93.26(3.22)**
Sonar	81.84(5.67)	79.92(5.16)	**82.06(2.09)**
Ionosphere	**89.06(2.21)**	89.06(2.55)	89.06(2.21)
Wine	**98.53(1.48)**	**98.53(1.08)**	**98.53(1.08)**
Haberman	76.13(2.06)	**76.42(2.38)**	76.13(1.56)
CTG	**86.04(1.14)**	85.99(1.05)	**86.08(1.24)**
Vertebral	85.62(2.35)	**85.76(2.55)**	84.90(2.95)
Faults	68.72(1.19)	68.63(1.24)	**68.95(1.04)**
WDVG1	**84.84(0.36)**	84.83(0.63)	84.77(0.65)
Ecoli	**80.92(3.76)**	80.66(3.58)	80.66(3.48)
GLASS	65.21(3.65)	**66.04(3.67)**	65.21(3.53)
ILPD	70.17(2.33)	**70.48(2.28)**	69.64(2.47)
Adult	87.22(1.84)	87.29(2.20)	**87.29(1.80)**
Weaning	79.69(3.71)	79.83(2.94)	**79.98(3.55)**
Laryngeal1	87.00(5.00)	86.79(4.72)	**87.21(5.35)**
Laryngeal3	73.42(1.26)	**73.79(1.38)**	73.54(1.66)
Thyroid	97.38(0.67)	**97.44(0.71)**	97.38(0.67)
German	74.54(1.30)	**75.03(2.04)**	74.36(1.28)
Heart	85.30(2.30)	**85.46(2.70)**	85.46(2.70)
Segmentation	96.42(0.76)	96.34(0.74)	**96.46(0.79)**
Phoneme	81.77(0.72)	81.47(0.77)	**81.82(0.69)**
Monk2	83.34(3.32)	82.83(3.82)	**83.45(3.46)**
Mammographic	84.41(2.54)	**84.62(2.46)**	84.30(2.27)
Magic Gamma Telescope	85.33(2.29)	84.62(2.46)	**85.65(2.27)**
Friedman Average Rank (↓)	2.15	1.98	**1.86**

3.5 Comparison with the state-of-the-art DES techniques

In this section, we compare the recognition rates obtained by the proposed META-DES.H against eight state-of-the-art dynamic selection techniques in the DES literature: the KNORA-ELIMINATE [14], KNORA-UNION [14], DES-FA [20], Local Classifier Accuracy (LCA) [22], Overall Local Accuracy (OLA) [22], Modified Local Accuracy (MLA) [29], Multiple Classifier Behaviour (MCB) [21] and K-Nearests Output Profiles (KNOP) [16].

For all techniques, we use the same pool of classifiers defined in the previous section (Section 3.3) in order to have a fair comparison. The size of the region of competence (neighborhood size), K is set to 7 since it achieved the best result in previous experiments [1; 20]. The comparative results are shown in Table II-4. Due to size constraints, we only show the results using Naive Bayes as the meta-classifier since it achieved the highest recognition accuracy in the previous experiment. For each dataset, a Kruskal-Wallis statistical test with 95% confidence was conducted to know if the classification improvement is statistically significant. Results that are statistically better are marked with a •. The results of the proposed technique obtained the highest accuracy in 20 out of 30 datasets. In addition, the accuracy of the proposed system was statistically superior in 15 out of 30 datasets. The original META-DES framework [2], without the improvements proposed in this paper, achieved results that are statistically superior in 10 out of the 30 datasets when compared with the state-of-the-art DES techniques.

Furthermore, we also consider the Wilcoxon test with 95% confidence, for a pairwise comparison between the classification performances of the proposed system against the performance of the state-of-the-art DES techniques over multiple datasets. The results of the Wilcoxon test are shown in the last row of the table. The performance of the proposed META-DES.H system is statistically better when all 30 datasets are considered. Hence, the experimental results demonstrate that the changes proposed in this paper lead to a significant gains in performance when compared to other DES algorithms.

4. Conclusion

Table-A II-4 Mean and standard deviation results of the accuracy obtained for the proposed META-DES using a Naive Bayes classifier for the meta-classifier λ and the hybrid combination approach. The best results are in bold. Results that are significantly better are marked with a •

Database	META-DES.H	KNORA-E	KNORA-U	DES-FA	LCA	OLA	MLA	MCB	KNOP
Pima	**77.93(1.86)**	73.79(1.86)	76.60(2.18)	73.95(1.61)	73.95(2.98)	73.95(2.56)	77.08(4.56)	76.56(3.71)	73.42(2.11)
Liver Disorders	**69.95(3.49)** •	56.65(3.28)	56.97(3.76)	61.62(3.81)	58.13(4.01)	58.13(3.27)	58.00(4.25)	58.00(4.25)	65.23(2.29)
Breast (WDBC)	97.25(0.47)	97.59(1.10)	97.18(1.02)	**97.88(0.78)**	97.88(1.58)	97.88(1.58)	95.77(2.38)	97.18(1.38)	95.42(0.89)
Blood Transfusion	**78.25(1.37)** •	77.65(3.62)	77.12(3.36)	73.40(1.16)	75.00(2.87)	75.00(2.36)	76.06(2.68)	73.40(4.19)	77.54(2.03)
Banana	94.51(2.36)	93.08(1.67)	92.28(2.87)	**95.21(3.18)**	95.21(2.15)	95.21(2.15)	80.31(7.20)	88.29(3.38)	90.73(3.45)
Vehicle	83.55(2.10)	83.01(1.54)	82.54(1.70)	82.54(4.05)	80.33(1.84)	81.50(3.24)	74.05(6.65)	**84.90(2.01)**	80.09(1.47)
Lithuanian Classes	93.26(3.22)	93.33(2.50)	95.33(2.64)	98.00(2.46)	85.71(2.20)	**98.66(3.85)**	88.33(3.89)	86.00(3.33)	89.33(2.29)
Sonar	**82.06(2.09)** •	74.95(2.79)	76.69(1.94)	78.52(3.86)	76.51(2.06)	74.52(1.54)	76.91(3.20)	76.56(2.58)	75.72(2.82)
Ionosphere	89.06(2.21)	**89.77(3.07)**	87.50(1.67)	88.63(2.12)	88.00(1.98)	88.63(1.98)	81.81(2.52)	87.50(2.15)	85.71(5.52)
Wine	**98.53(1.08)** •	97.77(1.53)	97.77(1.62)	95.55(1.77)	85.71(2.25)	88.88(3.02)	88.88(3.02)	97.77(1.62)	95.50(4.14)
Haberman	**76.13(1.56)** •	71.23(4.16)	73.68(2.27)	72.36(2.41)	70.16(3.56)	69.73(4.17)	73.68(3.61)	67.10(7.65)	75.00(3.40)
Cardiotocography (CTG)	86.08(1.24)	**86.27(1.57)**	85.71(2.20)	**86.27(1.57)**	86.65(2.35)	86.65(2.35)	86.27(1.78)	85.71(2.21)	86.02(3.04)
Vertebral Column	84.90(2.95)	85.89(2.27)	**87.17(2.24)**	82.05(3.20)	85.00(3.25)	85.89(3.74)	77.94(5.80)	84.61(3.95)	86.98(3.21)
Steel Plate Faults	**68.95(1.04)**	67.35(2.01)	67.96(1.98)	68.17(1.59)	66.00(1.69)	66.52(1.65)	67.76(1.54)	68.17(1.59)	68.57(1.85)
WDG V1	**84.77(0.65)** •	84.01(1.10)	84.01(1.10)	84.01(1.10)	80.50(0.56)	80.50(0.56)	79.95(0.85)	78.75(1.35)	84.21(0.45)
Ecoli	**80.66(3.48)**	76.47(2.76)	75.29(3.41)	75.29(3.41)	75.29(3.41)	75.29(3.41)	76.47(3.06)	76.47(3.06)	80.00(4.25)
Glass	65.21(3.53)	57.65(5.85)	61.00(2.88)	55.32(4.98)	59.45(2.65)	57.60(3.65)	57.60(3.65)	**67.92(3.24)**	62.45(3.65)
ILPD	**69.64(2.47)**	67.12(2.35)	69.17(1.58)	67.12(2.35)	69.86(2.20)	69.86(2.20)	69.86(2.20)	68.49(3.27)	68.49(3.27)
Adult	**87.29(1.80)** •	80.34(1.57)	79.76(2.26)	80.34(1.57)	83.58(2.32)	82.08(2.42)	80.34(1.32)	78.61(3.32)	79.76(2.26)
Weaning	**79.98(3.55)**	78.94(1.25)	81.57(3.65)	82.89(3.52)	77.63(2.35)	77.63(2.35)	80.26(1.52)	81.57(2.86)	82.57(3.33)
Laryngeal1	**87.21(5.35)** •	77.35(4.45)	77.35(4.45)	77.35(4.45)	77.35(4.45)	77.35(4.45)	75.47(5.55)	77.35(4.45)	77.35(4.45)
Laryngeal3	**73.54(1.66)**	70.78(3.68)	72.03(1.89)	72.03(1.89)	72.90(2.30)	71.91(1.01)	61.79(7.80)	71.91(1.01)	73.03(1.89)
Thyroid	**97.38(0.67)** •	95.95(1.25)	95.95(1.25)	95.37(2.02)	95.95(1.25)	95.95(1.25)	94.79(2.30)	95.95(1.25)	95.95(1.25)
German credit	**74.54(0.30)** •	72.80(1.95)	72.40(1.80)	74.00(3.30)	73.33(2.85)	71.20(2.52)	71.20(2.52)	73.60(3.30)	73.60(3.30)
Heart	85.46(2.70)	83.82(4.05)	83.82(4.05)	83.82(4.05)	85.29(3.69)	85.29(3.69)	**86.76(5.50)**	83.82(4.05)	83.82(4.05)
Satimage	**96.72(0.76)** •	95.35(1.23)	95.86(1.07)	93.00(2.90)	95.00(1.40)	94.14(1.07)	93.28(2.10)	95.86(1.07)	95.86(1.07)
Phoneme	**81.82(0.69)** •	79.06(2.50)	78.92(3.33)	79.06(2.50)	78.84(2.53)	78.84(2.53)	64.94(7.75)	73.37(5.55)	78.92(3.33)
Monk2	**83.45(3.46)** •	80.55(3.32)	77.77(4.25)	75.92(4.25)	74.07(6.60)	74.07(6.60)	75.92(5.65)	74.07(6.60)	80.55(3.32)
Mammographic	**84.30(2.27)** •	82.21(2.27)	82.21(2.27)	80.28(3.02)	82.21(2.27	82.21(2.27)	75.55(5.50)	81.25(2.07)	82.21(2.27)
MAGIC Gamma Telescope	**85.65(2.27)** •	80.03(3.25)	79.99(3.55)	81.73(3.27)	81.53(3.35)	81.16(3.00)	73.13(6.35)	75.91(5.35)	80.03(3.25)
Wilcoxon Signed test	n/a	$-$ ($\rho = .0001$)	$-$ ($\rho = .0007$)	$-$ ($\rho = .0016$)	$-$ ($\rho = .0001$)	$-$ ($\rho = .0001$)	$-$ ($\rho = .0001$)	$-$ ($\rho = .0003$)	$-$ ($\rho = .005$)

In this paper, we proposed two modifications to the novel META-DES framework. First, we compared different classifier models, such as the MLP Neural Network, Support Vector Machines with Gaussian Kernel (SVM), Random Forests and Naive Bayes for the meta-classifier. Next, we evaluated three combination approaches to the framework: Dynamic selection, Dynamic weighting and Hybrid. In the Dynamic selection approach, only the classifiers that attain a certain level of competence are used to classify a given query sample. In the dynamic weighting approach, all base classifiers in the pool are considered to give the final decision, with the meta-classifier estimating the weight of each base classifier. In the hybrid approach, only the classifiers that attain a certain level of competence are initially selected, after which their decisions are aggregated in a weighted majority voting scheme. Thus, the base classifiers attaining higher levels of competence have a greater impact on the final decision.

Experiments were conducted using 30 classification datasets derived from five different data repositories (UCI, KEEL, STATLOG, LKC and ELENA). First, we observed a significant im-

provement in accuracy using different classifier models for the meta-problem. The performance of the META-DES trained using a Naive Bayes for the meta-classifier achieves results that are statistically better compared to those achieved using an MLP Neural Network, according to the Wilcoxon Signed Rank test with 95% confidence. This finding confirms the initial hypothesis that the overall performance of the system improves when the recognition accuracy of the meta-classifier improves. As the META-DES framework considers the dynamic selection problem as a meta-classification problem, we can improve the recognition accuracy by focusing only on improving the classification performance in the meta-problem. This finding is especially useful for ill-defined problems since there is not enough data to properly train the base classifiers. Techniques such as stacked generalization for the generation of more meta-feature vectors in the data generation process as well as the use of feature selection techniques to achieve a more representative set of meta-features can be considered to improve the recognition performance at the meta-classification level.

In addition, we demonstrate that the framework can also be used to compute the weights of the base classifiers. We found that the Naive Bayes classifier achieved the best result when the dynamic weighting (META-DES.W) or hybrid (META-DES.H) approach is used. This can be explained by the fact that the supports given by this classifier can be seen as the likelihood that the base classifier belongs to the "‘competent"’ meta-class. Thus, the classifiers that are more likely to be "‘competent"’ have greater influence on the classification of any given test sample. When compared to eight state-of-the-art techniques found in the dynamic ensemble selection literature, the proposed META-DES.H using a Naive Bayes classifier for the meta-classifier presented classification accuracy that is statistically better in 15 out of the 30 classification datasets. The original META-DES framework [2] achieved results that are statistically better in 10 out of the 30 datasets when compared with the state-of-the-art DES techniques. Hence, the changes to the META-DES framework proposed in this paper lead to a significant gain in performance when compared against other DES algorithms.

APPENDIX III

FEATURE REPRESENTATION SELECTION BASED ON CLASSIFIER PROJECTION SPACE AND ORACLE ANALYSIS

Rafael M. O. Cruz[1], George D. C. Cavalcanti[2], Tsang Ing Ren[2], Robert Sabourin[1]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Published in « Expert Systems with Applications » 2013.

Abstract

One of the main problems in pattern recognition is obtaining the best set of features to represent the data. In recent years, several feature extraction algorithms have been proposed. However, due to the high degree of variability of the patterns, it is difficult to design a single representation that can capture the complex structure of the data. One possible solution to this problem is to use a multiple-classifier system (MCS) based on multiple feature representations. Unfortunately, still missing in the literature is a methodology for comparing and selecting feature extraction techniques based on the dissimilarity of the feature representations. In this paper, we propose a framework based on dissimilarity metrics and the intersection of errors, in order to analyze the relationships among feature representations. Each representation is used to train a classifier, and the results are compared by means of a dissimilarity metric. Then, with the aid of Multidimensional Scaling, visual representations are obtained of each of the dissimilarities and used as a guide to identify those that are either complementary or redundant. We applied the proposed framework to the problem of handwritten character and digit recognition. The analysis is followed by the use of an MCS built on the assumption that combining dissimilar feature representations can greatly improve the performance of the system. Experimental results demonstrate that a significant improvement in classification accuracy is achieved due to the complementary nature of the representations. Moreover, the proposed MCS obtained the

best results to date for both the MNIST handwritten digit dataset and the Cursive Character Challenge (C-CUBE) dataset.

1. Introduction

The selection of the feature extraction algorithm is known to be an important factor in the performance of any recognition system [108]. However, designing a single feature extraction algorithm in a complex recognition problem that can recognize every kind of pattern is unlikely, because of the high degree of variability of the data. Some features might present a better result for a predetermined class of patterns. For instance, in the problem of handwritten character recognition, one feature extraction algorithm might represent lowercase letters better, while another is a more robust performer for uppercase letters. Moreover, every feature extraction technique represents a different aspect of the image, such as concavities [109], character structure [110], edges [111], projections [111], and directional information based on the gradient [112].

In our opinion, the information captured by different feature extraction techniques can be complementary, and a multiple-classifier system (MCS) developed using multiple feature representations achieves higher classification performance. Unfortunately, there is no framework in the literature for comparing and analyzing the relationships among feature representations. Feature extraction techniques are only compared based on classification accuracy, and none analyzes the diversity among them.

In the MCS context, the system can only perform better than the best individual classifier when there is diversity among the classifiers [8], so that they achieve different solutions. In other words, we seek significantly different representations because they produce different solutions - combining techniques that perform identically is not useful.

In this paper, we propose a novel framework to study the relationships among the various feature representations. Each feature extraction technique is used to train a classifier. Their results are evaluated based on dissimilarity/diversity measures [8; 50]. Then, the relationships ob-

tained are used to project each representation onto a space (Classifier Projection Space [113]). Each feature extraction technique is represented by a point, and the distance between two points corresponds to the difference between them. In this way, a spatial relationship is achieved between different representations. Feature representations that are close to one another produce similar results, and so may be redundant. Combining them is unlikely to improve the accuracy of the system, and they could be removed from the system without a significant loss in performance. Those that are far apart are able to correctly recognize different classes of images, and should be considered for an MCS.

The purpose of our proposed framework in this context is twofold: to perform an analysis of the complementarity within a subset of feature extraction methods, and to serve as a methodology for identifying and removing feature representations that produce similar results. In this way, a more efficient MCS is achieved.

We apply this framework to the problem of handwritten character and digit recognition. This is an important area in the field of pattern recognition, because of the many practical applications that exist, such as mail sorting, bank check analysis, and form processing, all of which depend on quality feature extraction techniques. Pattern recognition in handwritten documents is a major challenge, owing to the diversity of handwriting styles. A writer can, for example, change his writing style as a result of a change in his neurological status, the type of pen he uses, and his hand position [114], especially if the shapes of the characters are complex [115].

A total of nine feature extraction techniques for handwritten recognition are evaluated here. Two of them, Modified Edge-Maps and Multi-Zoning, are based on classical algorithms. We selected techniques that capture different views of the image, such as concavities and projections, as well as techniques that capture the same type of information, such as directional information based on the gradient, but are extracted using different algorithms. Our analysis enables us to answer the following questions:

a. Do different feature extraction techniques present complementary information (i.e. are they able to correctly classify different images)?

b. Are feature extraction techniques that use a similar approach (e.g. different methods for extracting the gradient) less complementary than techniques that use different characteristics (e.g. edges, concavities)?

c. Can the proposed framework be used to select a subset of feature representations?

We perform an analysis of feature representations, which serves as the basis on which we propose a novel MCS for handwritten recognition. The proposed system is applied to two different handwritten recognition tasks: digit recognition, and cursive character recognition. For the handwritten digit recognition problem, we use the MNIST database, which is a very well-known benchmark. For cursive character recognition, we use the Cursive Character Challenge database (C-Cube). We carry out a sensitivity analysis for both cases, and demonstrate that the use of complementary feature representations greatly improves recognition performance. We also show that a scheme that includes a Multi-Layer Perceptron (MLP) neural network trained to combine the classifiers presented the highest accuracy rates in both cases, and these rates are also the best results obtained for these databases to date.

This paper is organized as follows. The framework for feature representation analysis is introduced in Section 2. Section 3 describes the nine feature extraction techniques studied in this paper. The evaluation of each feature extraction algorithm and the sensitivity analysis of these algorithms are shown in Section 4. Section 5 shows the performance of the system when an MCS is designed based on different feature representations. Finally, our conclusion is presented in the last section.

2. Feature Representation Analysis

This section describes the feature representation selection scheme shown in Figure III-1. The first step in this approach is to extract m different feature representations, $F_1, \ldots, F_m$, of the patterns from the data (DB). These feature representations are used to train m classifiers, $C_1, \ldots, C_m$, separately. Then, the dissimilarity matrix D (Section 2.1) and its projection onto the classifier space $\tilde{D}$ (Section 2.2) are computed. The matrix $\tilde{D}$ contains the spatial relation-

ship between the classifiers that is equivalent to their dissimilarities (matrix D). That spatial relationship is used to perform the sensitivity analysis (Section 2.3), from which redundant representations can be identified. Finally, a subset $m' \subset m$ of the feature representations is selected.

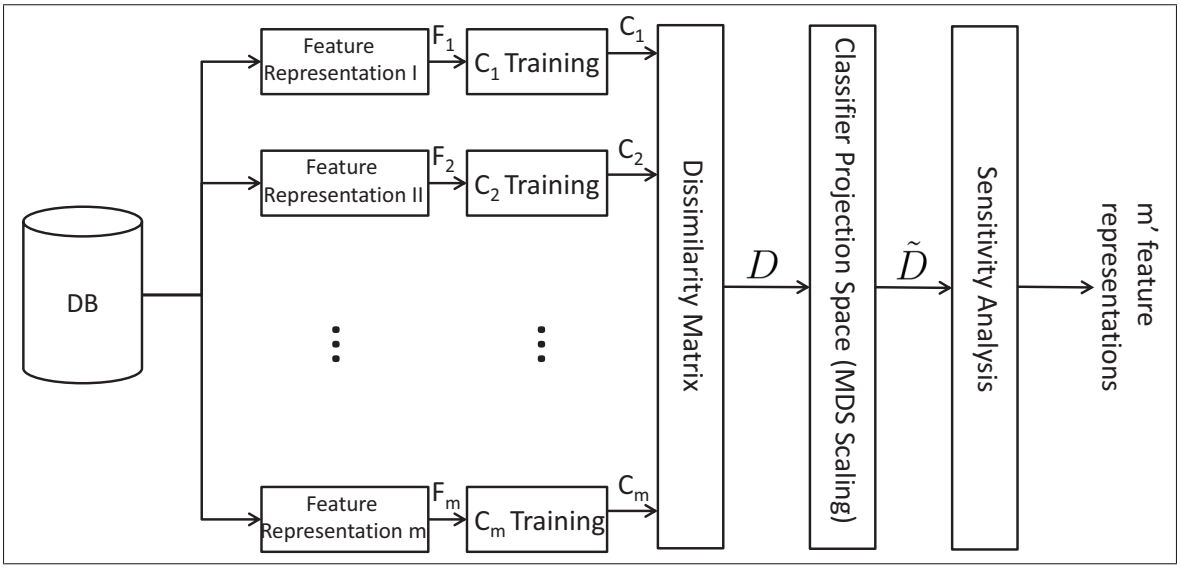


Figure-A III-1 Overview of the proposed feature representation selection scheme. Each F_i represents one feature representation, and it is used to train the classifier C_i. Based on the output of each pair (feature representation, classifier), we compute the dissimilarity matrix D that is used to perform the classifier projection $\tilde{D}$ through multidimensional scaling (MDS). The matrix $\tilde{D}$ is used to analyze the complementarity of the feature representations and perform the selection.

2.1 Dissimilarity Matrix

The matrix D is an $m \times m$ symmetrical matrix, where each member $d(i, j)$ represents the dissimilarity between the classifiers C_i and C_j. In order to compute D, we first need to select an appropriate metric that measures the difference between feature representations. There are many diversity measures in the literature [8]. We selected the Double Fault [50], because it has already been demonstrated that this measure presents a positive correlation with ensemble accuracy [116]. Equation A III-1 shows the Double Fault measure between a pair of classifiers C_i and C_j.

$$d(i,j) = \frac{N^{00}}{N^{00} + N^{01} + N^{10} + N^{11}} \quad \text{(A III-1)}$$

where N^{ij} is the number of examples correctly classified (1) or misclassified (0) for the classifiers C_i and C_j respectively. In other words, the Double Fault measures the probability that the same pattern is misclassified by both classifiers.

2.2 Classifier Projection Space

After the dissimilarity matrix D has been obtained, the next step is to project each feature representation onto the Classifier Projection Space (CPS). The CPS is a $\mathbb{R}^{\kappa}$ space, where each classifier is represented as a point, and the Euclidean distance between two classifiers represents their dissimilarity [113]. Classifiers that are similar are closer together in the CPS, while those that are less similar are further apart. In this way, it is possible using CPS to obtain the spatial representations of all the classifiers. This spatial representation provides a better understanding of the relationships among the classifiers than when only the value of the diversity measure is used. The diversity measure only describes the relationship between a pair of classifiers, while the CPS shows the relationships among all the classifiers. A two-dimensional CPS is used for better visualization. In order to obtain a two-dimensional classifier projection, a dimensionality reduction of the data is required. This can be achieved using Multidimensional Scaling (MDS) [113; 117], which refers to a group of methods used to visualize high-dimensional data mapped to a lower dimensional space [118].

Given the dissimilarity matrix D, a configuration X of m points in $\mathbb{R}^k, (k \leq m)$ is computed using a linear mapping, called classical scaling [117]. The process is performed through rotation and translation, such that the distances after dimensionality reduction are preserved. The projection X is computed as follows: first, a matrix of the inner products is obtained by the square distances $B = -\frac{1}{2} J D^2 J$, where $J = I - \frac{1}{m} U U^T$, and I and U are the identity matrix and unit matrix respectively. J is used as a normalization matrix, so that the mean of the data is zero. The eigendecomposition of B is then obtained, $B = Q \Lambda Q^T$, where Λ is a diagonal ma-

trix containing the eigenvalues (in decreasing order) and Q is the matrix of the corresponding eigenvectors. The configuration of points in the reduced space is determined by the k largest eigenvalues. Therefore, X is uncorrelated in the space $\mathbb{R}^k$, $X = Q_k\sqrt{\Lambda_k}$. In our case, $k = 2$.

MDS is obtained by applying the Sammon mapping over X. The Sammon mapping is a non-linear projection that preserves the distances between the points [113; 117]. The mapping is performed by defining a function, called stress function $\mathcal{S}$ (Equation A III-2), which measures the difference between the original dissimilarity matrix D and the distance matrix of the projected configuration, $\tilde{D}$, where $\tilde{d}(i,j)$ is the distance between the classifiers i and j in the projection X, as defined in equation 2:

$$\mathcal{S} = \frac{1}{\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} d(i,j)^2} \sum_{i=1}^{m-1}\sum_{j=i+1}^{m} (d(i,j) - \tilde{d}(i,j)) \qquad \text{(A III-2)}$$

In other words, the objective of $\mathcal{S}$ is to minimize the difference between D and $\tilde{D}$, and so the projection onto the CPS is found in iterative fashion. The algorithm starts with an initial representation of points in Euclidean space (the configuration of points in X with its corresponding distance matrix $\tilde{D}$). Then, the configuration of the points is adjusted to minimize $\mathcal{S}$. A scaled gradient algorithm [113] is used for this purpose. In the end, the distances between the classifiers correspond to an approximation of their original dissimilarity.

Figure III-2 shows an example of the CPS space for different feature representations extracted from the Iris dataset[1]. This dataset consists of four features. In order to simulate different feature representations, we use random combinations of two and three features. Ten different representations were generated: FS I to FS VI are combinations of two features, FS VII to FS IX are combinations of three features, and FS X is a representation consisting of all four features. A Perceptron was used as the classifier for each feature representation.

[1] http://archive.ics.uci.edu/ml/

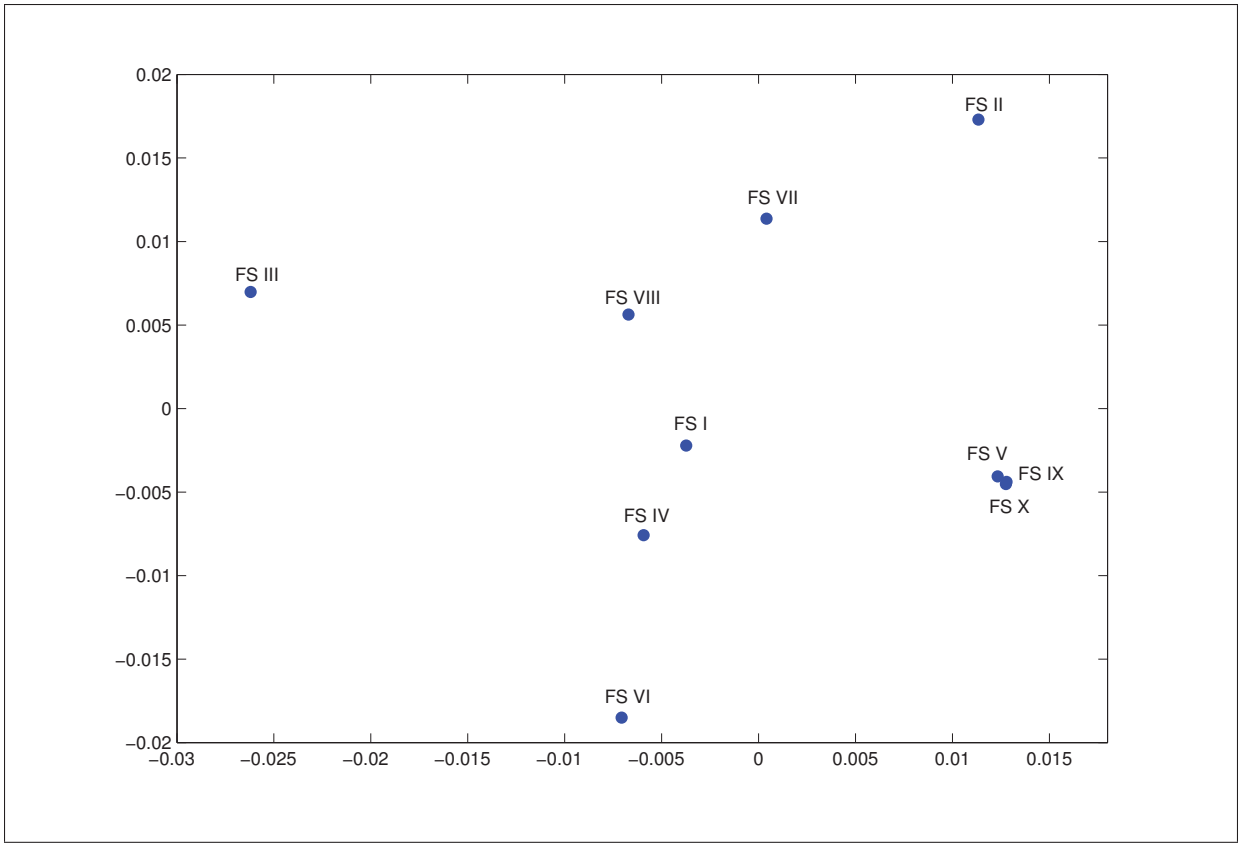


Figure-A III-2 Example of a two-dimensional CPS plot for different feature representations extracted from the iris dataset.

2.3 Sensitivity Analysis

The first step in the sensitivity analysis is to use the CPS as a visual tool to group feature representations based on the spatial information provided by the CPS. In Figure III-2, we can observe that there are three feature representations that are really close together: FS V, FS IX, and FS X. Consequently, they are probably redundant. In contrast, some feature representations, such as FS II and FS VI, are far apart, and can be considered to be from different groups. We can see that the CPS is used to identify groups of representations that perform in similar fashion.

The second step is to analyze the performance of some combinations of feature representations. This is achieved using the concept of the Oracle, which produces the best possible result of any

combination of classifiers [119]. It considers that the ensemble obtains the correct classification if at least one classifier produces the true label. So, based on the analysis of the error performed by Oracle, it is possible to know whether or not individual classifiers are able to correctly recognize different patterns.

From the analysis in Figure III-2, we constructed two diagrams. The first is composed of feature representations that are close together: FS V, FS IX, and FS X (Figure III-3a). The second is composed of representations that are far apart, and can be considered to belong to different groups: FS II, FS III, and FS VI (Figure III-3b). The number inside each circle indicates the number of errors committed by each classifier. The area where the classifiers intersect represents the errors committed by all of them, and can be viewed as the error obtained by the Oracle combination (i.e. (FS V $\cap$ FS IX) is the number of patterns misclassified by the Oracle combination).

The total number of errors obtained using FS X is 28 (Figure III-3a). However, none of these errors was committed by this feature representation alone (i.e. an individual error). The majority of the errors lie at the intersection of the three feature spaces. In other words, 23 patterns are misclassified in the three feature subspaces, while 4 and 2 are common errors obtained by the intersections (FS X $\cap$ FS IX) and (FS X $\cap$ FS V) respectively. So, errors committed by classifiers in the same group are likely to occur in the same patterns, which means that combining them is unlikely to improve recognition performance.

In contrast, when representations that are far apart are combined (i.e. they belong to different groups, such as FS II, FS III, and FS VI) (Figure III-3b), we can observe that the intersection of the errors produces a lower value. For instance, from the 19 errors committed by FS III, 16 occur individually. The intersections (FS III $\cap$ FS II) and (FS III $\cap$ FS VI) produce errors of 1 and 2 respectively. Moreover, looking at the intersection of the three techniques, we note that no pattern was misclassified by all the techniques, as opposed to 23 in Figure III-3a. Therefore, these feature representations can be considered complementary, since they can correctly recognize different patterns.

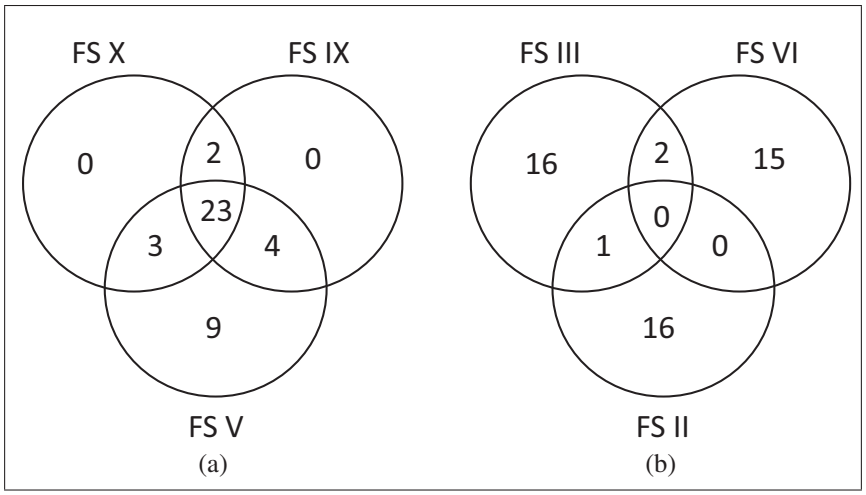


Figure-A III-3 Oracle error analysis for different feature representations trained on the Iris dataset. (a) representations that belong to the same cluster. (b) representations that belong to different clusters.

Consequently, we can identify representations that are redundant using the sensitivity analysis. From the point of view of an MCS, there is no advantage to combining FS V with FS IX and FS X for the Iris dataset, as they behave almost identically. So, instead of using m representations, we can use the sensitivity analysis to select a more efficient set $m' = 3$ (FS II, FS III, and FS VI) that consists only of dissimilar representations. We expect that the subset m' produces results that are better than, or at least comparable to, the whole set m. Applying this methodology in the context of feature representations, it is possible to compare different representations and select only the most dissimilar ones. The selected feature representation is used to construct a more robust MCS for pattern recognition problems.

3. Feature Extraction Methods

Feature extraction can be defined as a means for obtaining the most relevant information to be used in the classification procedure [120]. There are several feature extraction techniques, and choosing a technique can be considered the most important factor in the achievement of high accuracy rates in a pattern recognition problem [108]. A total of nine feature extraction

algorithms are summarized below. Feature sets I to VII have been proposed by others [110; 111; 109; 121; 122], and feature sets VIII and IX are new contributions.

3.1 Feature Set I: Structural Characteristics

This feature set is obtained by combining projections and profiles in a single feature vector. First, the input image is scaled to a 32×32 matrix. Then, three types of histogram (horizontal, vertical, and radial) and two types of profile (radial in-out and radial out-in) are computed.

The horizontal and vertical histograms (Figure III-4b and Figure III-4c) are calculated by summing the number of black pixels in each line and column respectively. So, 32 features are generated for each histogram.

The radial histogram (Figure III-4d) is computed as the number of black pixels in 72 directions at 5 degree intervals. The process progresses from the centroid of the image to its border, and 72 features are generated.

Radial In-Out and Radial Out-In profiles are defined by the position of the first and last black pixel respectively, from a search that progresses from the centroid of the image to its border in 72 directions at 5 degree intervals. In this way, each profile generates 72 features. These features form a 280-dimensional feature vector (32 horizontal projections + 32 vertical projections + 72 radial projections + 72 In-Out profiles + 72 Out-In profiles). Details of this technique are described in [110].

3.2 Feature Set II: Image Projections

This method consists of extracting the radial and diagonal projections. The diagonal projections are computed by grouping the pixels in two diagonal lines (45° and -45°). A total of 32 features are obtained for each diagonal.

To extract the radial projections, the image must first be divided into four quadrants: top, bottom, right, and left. The quadrants are used to remove rotational invariance, which is an

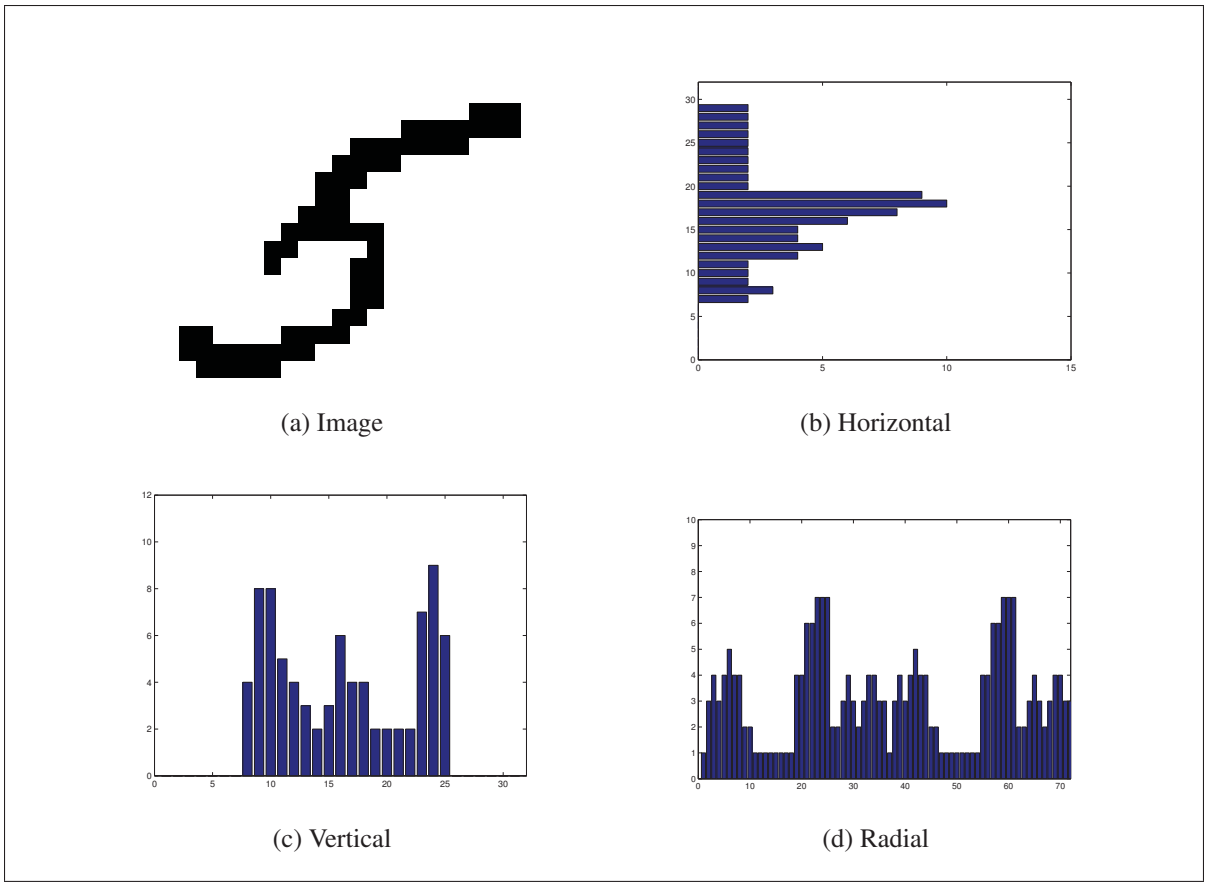


Figure-A III-4 Feature Set I: Example of a 5 digit projections.

undesirable characteristic in handwritten recognition, since it makes it impossible to distinguish between some digits (e.g. digits 6 and 9).

For each quadrant, the radial projections are obtained by grouping pixels from its radial distance to the centroid of the image. The values of each projection are normalized to a $[0-1]$ range. The normalized features are combined into a single vector containing 128 features (16 for each radial projection and 32 for each diagonal projection). More details about this procedure are described in [111].

3.3 Feature Set III: Concavity Measurement

These features are obtained using the following steps: The image (Figure III-5c) is scaled to a matrix 18×15, and divided into six zones. Each part contains its own 13-dimensional feature vector, and the position of each feature vector corresponds to one of the 13 possible configurations (Figure III-5d).

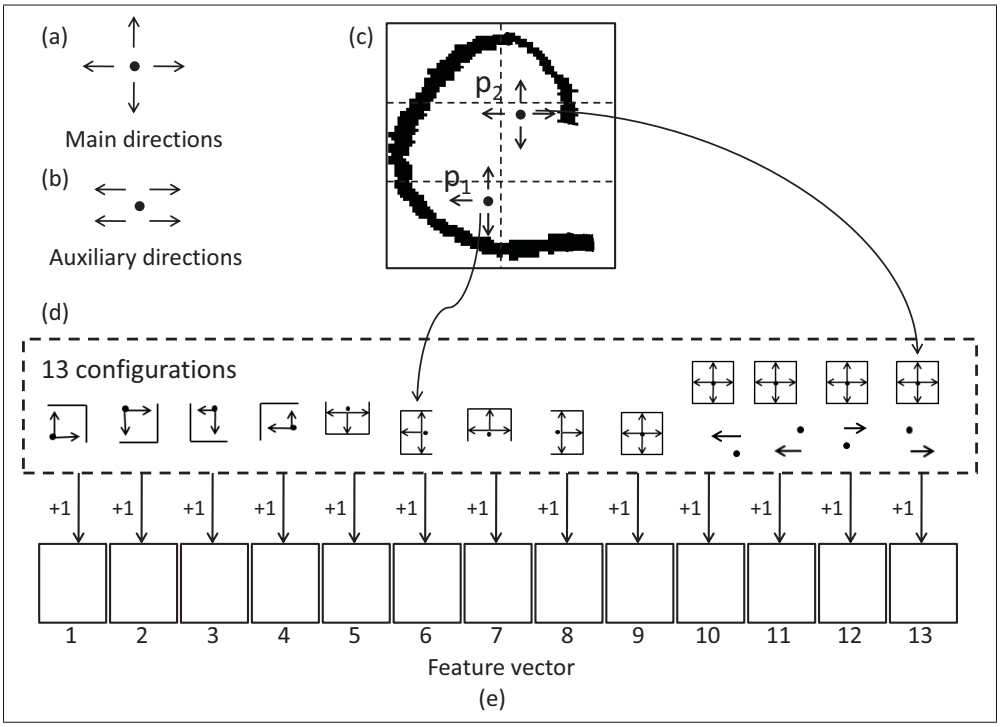


Figure-A III-5 Feature Set V: The Concavity Measurement procedure (a) Main directions. (b) Auxiliary directions. (c) Query image. (d) The thirteen possible configurations. (e) Feature vector.

For each white pixel (background), the algorithm conducts a search, starting from that pixel and moving in each of the four "main directions" (Figure III-5a). The search continues until a black pixel (foreground) is found, or when the end of the image is reached. Finally, the number of directions ending with a black pixel is computed, as are the directions themselves, each of which corresponds to one of the 13 possible configurations (Figure III-5d). So, the configuration in the feature vector corresponding to the result of the search is incremented.

However, in some cases, the search may find a black pixel in the four "main directions", but that pixel is not in a closed area. In order to guarantee that the white pixel is in a closed region, a new search is performed using the "auxiliary directions" (Figure III-5b). If the search using one of the auxiliary directions reaches the end of the image without finding a black pixel, the correct configuration (from the 10th to the 13th) is incremented. Otherwise, the point is in a closed region (9th position of the feature vector).

To better understand the method, we analyze two cases. In the case of P_1, the search finds a black pixel in three directions: top, bottom, and left. So, the configuration corresponds to the 6th position of the vector (Figure III-5e), and this position is incremented. In the case of P_2, the search in the four main directions finds a black pixel. However, using the auxiliary directions, the search also finds that the point is not in a closed region (no black pixel was found in the bottom right auxiliary direction). Therefore, P_2 corresponds to the 13th configuration.

These steps are computed for the six zones separately. At the end of the process, the feature vectors of each zone are combined into a single vector with 78 (13×6) features. A detailed description of the algorithm is presented by Oliveira et al. [109].

3.4 Feature Set IV: MAT-based Directional Gradient

This algorithm computes the gradient components of a grayscale image. So, the first step in this procedure is to transform a binary image into a pseudo-grayscale image using the Medial Axial Transformation (MAT) algorithm. The Sobel operators in the horizontal S_x and vertical S_y directions are applied to the pseudo-grayscale image I_m, generating the X-gradient image I_{m_x} and the Y-gradient image I_{m_y}. These are defined as:

$$I_{m_x} = I_m * S_x \tag{A III-3}$$

$$I_{m_y} = I_m * S_y \tag{A III-4}$$

For each pixel, the magnitude $r(i,j)$ and the phase $\Theta(i,j)$ are defined as:

$$r(i,j) = \sqrt{I_{m_x}^2(i,j) + I_{m_y}^2(i,j)} \tag{A III-5}$$

$$\Theta(i,j) = \tan^{-1} \frac{I_{m_y}^2(i,j)}{I_{m_x}^2(i,j)} \tag{A III-6}$$

In order to generate a fixed number of features, the phase of each pixel $\Theta(i,j)$ is quantized into eight directions at $\pi/4$ intervals each. Then, the image is divided into 16 equally spaced sub images, and, for each sub image, the number of pixels in each of the eight directions is used as a feature. So, the feature vector size is equal to 128 (16 sub images $\times$ 8 directions). Details of this feature extraction algorithm can be found in [121].

3.5 Feature Set V: Binary Directional Gradient

This algorithm computes the gradient components of a binary image. The gradient is computed using the same procedure as that of a MAT-based directional gradient, defined in Section 3.4, except that no MAT transform is needed, because a binary image is used instead of a grayscale one. A total of 128 features are extracted per image.

3.6 Feature Set VI: Median Gradient

In this technique, the image is first enhanced using a median filter to remove noise. Next, the Robert operators [123] in the horizontal R_x and vertical R_y directions are applied to the filtered image to generate the X-gradient image I_{m_x} and the Y-gradient image I_{m_y}.

$$I_{m_x} = I_m * R_x \tag{A III-7}$$

$$I_{m_y} = I_m * R_y \qquad \text{(A III-8)}$$

The gradient is computed using the same procedure as described in the 3.4 section, generating 128 features. This method is described in detail by Zhang et al. [121].

3.7 Feature Set VII: Camastra 34D

This feature extraction algorithm was proposed by Camastra [122]. The image is divided into 16 sub images (cells), forming a 4×4 grid with a small overlap between them. Two operators are computed for each cell. The first is similar to the Zoning algorithm, and computes the number of black pixels (foreground) relative to the total number of black pixels in the whole image. The difference is that, in the Zoning algorithm, the number of black pixels is computed relative to the number of pixels in each zone. The second is a directional operator, which estimates the directions of the pixels. The method defines N equally spaced lines in the selected direction, after which the number of black pixels in each line is computed. The same steps are performed for the orthogonal direction. The difference between the selected direction and the orthogonal direction is used as a feature. The direction selected in this implementation was 0°, having the orthogonal direction of 90°. This results in a feature vector with 32 values. Two additional pieces of information were used as global features: The width/height ratio and the portion of the character that is below the baseline. The final vector consists of 34 features (16×2 local features $+$ 2 global features).

3.8 Proposed Feature Extraction Algorithms

3.8.1 Feature Set VIII: Multi-Zoning

The idea behind using multiple configurations of zones simultaneously is to compute information from the image at different levels of detail. Using larger zones, global information about the shape of the character can be computed. In smaller zones, the focus is on local details,

which are important for distinguishing between characters with similar shapes (e.g. digits 2 and 3). As a result, both global and local information is extracted at the same time.

This algorithm works as follows: an $M \times N$ image is divided into several sub images, and the percentage of black pixels in each sub image is used as feature. To achieve better recognition performance, many different divisions (Figure III-6) are selected and grouped to form the feature vector. A total of thirteen different configurations $(3\times1, 1\times3, 2\times3, 3\times2, 3\times3, 1\times4, 4\times1, 4\times4, 6\times1, 1\times6, 6\times2, 2\times6$, and $6\times6)$ were chosen, resulting in 123 $(3+3+6+6+9+4+4+16+6+6+18+18+36)$ features.

The Multi-Zoning technique differs from previous zoning techniques, such as the one described by Impedovo et al. [124], in that the latter use only one zoning configuration. In the proposed method, instead of searching for an optimal division, we use multiple divisions, in order to have a representation of the image at different levels of detail. Moreover, we expect to achieve a better result using multiple configurations, since it is difficult to find a single configuration that can deal with the high degree of variability among handwriting styles.

3.8.2 Feature Set IX: Modified Edge Maps

This algorithm is a modified version of the Edge Maps algorithm of Chin et al. [111]. An $M \times N$ image is first thinned using the Zhang-Suen algorithm [125] and scaled to a 25×25 matrix. Then, the Sobel operators [123] are used to extract four distinct edge maps: one horizontal, one vertical, and two diagonal (45° and -45°). Figure III-7 shows the four edge maps and the image after the thinning process has been performed.

The four edge maps and the thinned image are then divided into 25 sub images of 5×5 pixels each. The features are obtained through the computation of the percentage of black pixels in each sub image (25 features for each map). They are then combined to form a single feature vector containing 125 (25×5) features. The original algorithm, the Edge Maps algorithm of Chin et al. [111], does not compute the percentage of black pixels per sub image, but instead uses the value of each pixel in greyscale as features.

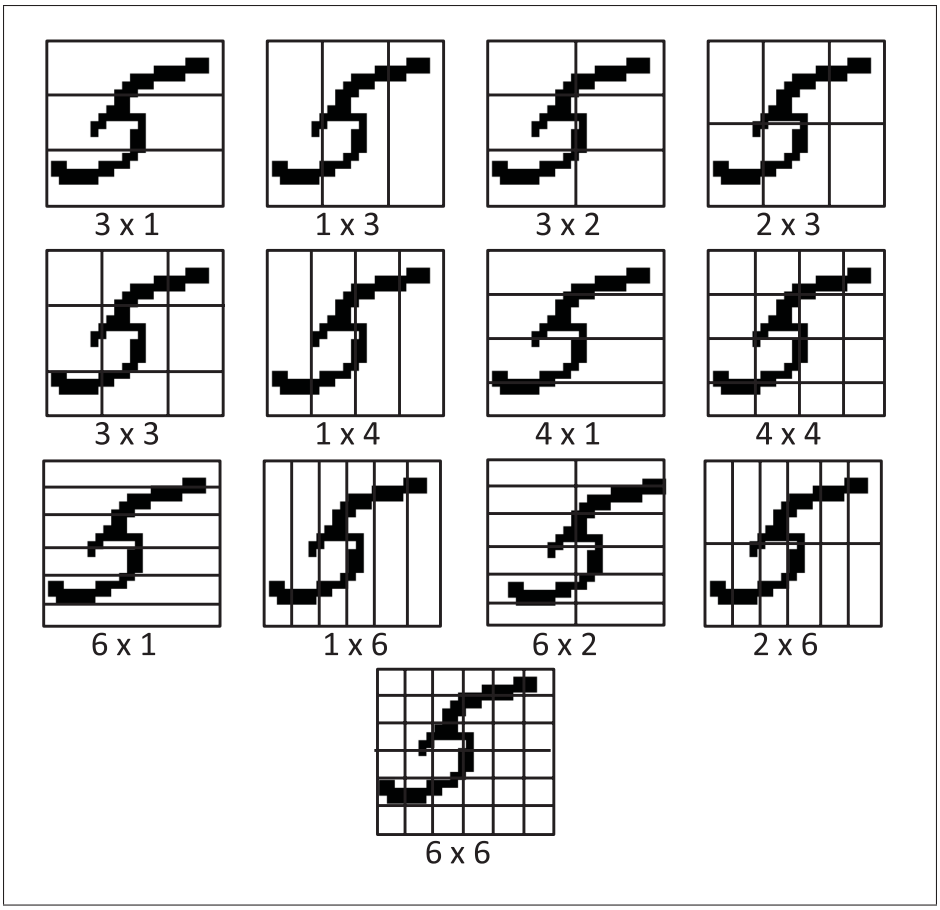


Figure-A III-6 Feature Set VIII: Thirteen configurations used in the Multi-Zoning technique.

4. Empirical Evaluation of Feature Extraction Techniques

The analysis of the feature extraction techniques was performed by conducting experiments using two different handwritten recognition problems: digit recognition and cursive character recognition. In the latter experiment, the Cursive Character Challenge database was used, while the handwritten digit recognition experiment was performed using the MNIST database. Both databases are publicly accessible, and both have been widely used as benchmarks.

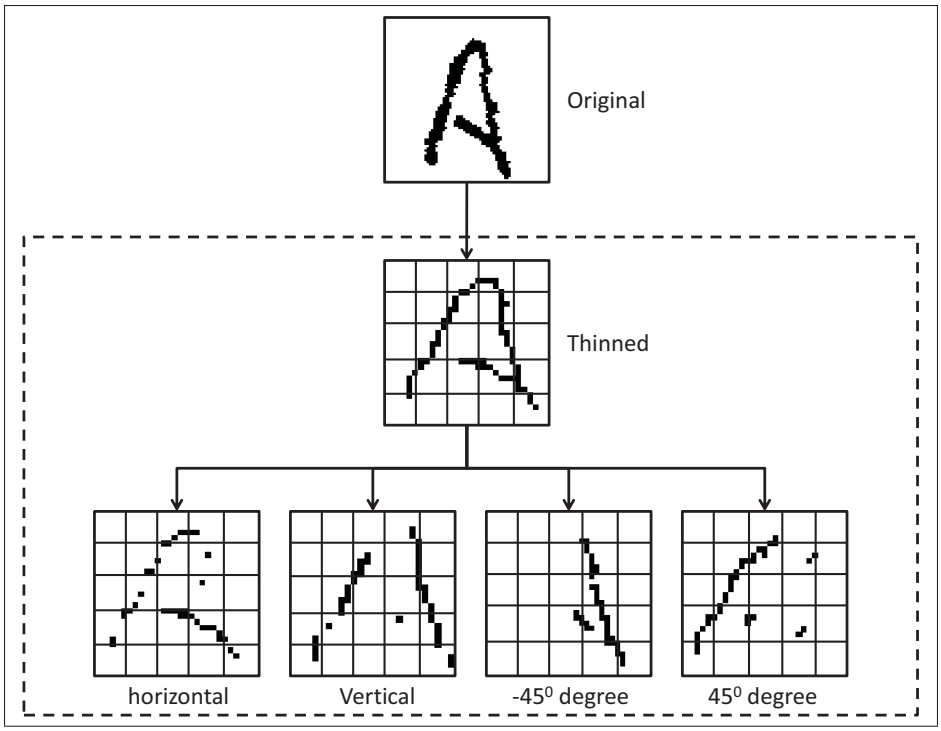


Figure-A III-7 Feature Set IX: Example of the process for obtaining the features for a character "A".

4.1 C-Cube Database

C-Cube is a public database available on the Cursive Character Challenge website [126]. It consists of 57,293 images, including both uppercase and lowercase letters, manually extracted from the CEDAR and United States Postal Service (USPS) databases. As reported by Camastra et al. [126], there are three advantages to using this database:

a. It is already divided into training sets and test sets, and so the results of different researchers can be rigorously compared.

b. It contains not only images, but also their feature vectors extracted using the algorithm proposed by Camastra [122].

c. The results obtained using the state-of-the-art methods still leave room for significant improvement.

The database is divided into 38,160 (22,274 lowercase and 15,886 uppercase) images for training, and 19,133 (11,161 lowercase and 7,972 uppercase) images for testing. All the images are binary and variable in size. For each image, four additional pieces of information are provided as global features: the distance between the base and the upper line, the distance between the upper extremity and the baseline, the distance between the lower extremity and the baseline, and the width/height ratio. The samples, which varied in number per class, were selected based on their frequency of occurrence in the documents extracted from the CEDAR and USPS datasets. Figures III-8 and III-9 show the distribution of the lowercase and uppercase letters respectively.

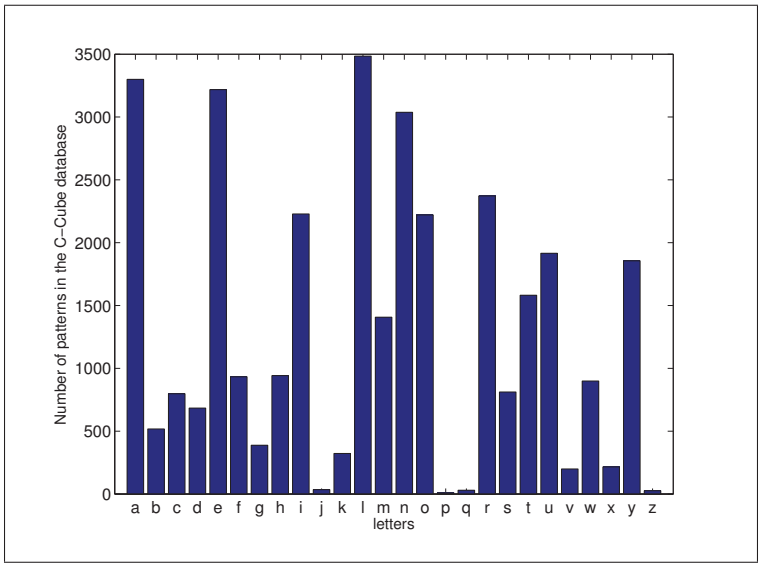


Figure-A III-8 Distribution of lowercase letters in the C-Cube Database.

Thornton et al. [127] observed that the image files (*test.chr* and *training.chr*) available on the C-Cube website do not match the feature vectors (*test.vec* and *training.vec*). For this reason, they labeled the dataset with the feature vectors as *Split A* and the dataset with the image files as *Split B*. In this work, only *Split B* is used, since the image files of the *Split A* are not available.

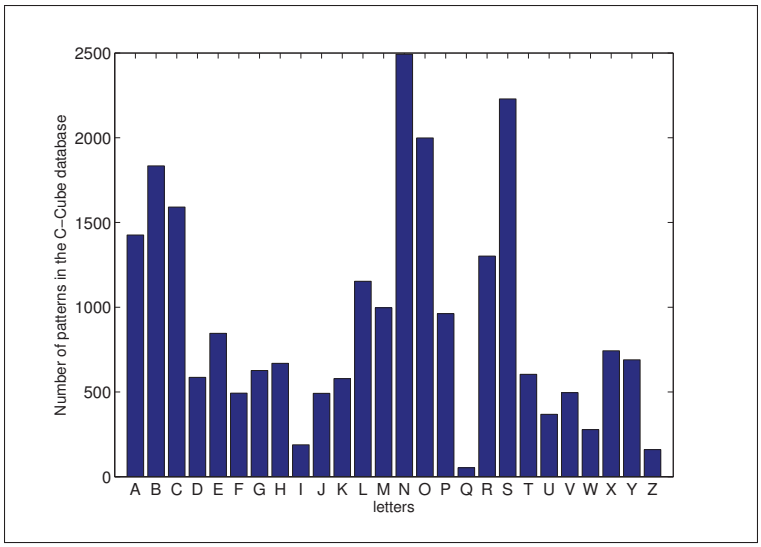


Figure-A III-9 Distribution of uppercase letters in the C-Cube Database.

4.2 MNIST Database

MNIST is a well-known handwritten digit recognition database. It contains 60,000 images for training and 10,000 images for testing. All the images in the dataset are size-normalized and centered to a 28×28 image.

The advantage of using this database is twofold. First, the images are already preprocessed. Second, the database is already divided into a test set and a training set. This makes it easy to compare the results obtained by different researchers.

4.3 Experimental Protocol

All the experiments were conducted using a three layer MLP, trained with the *Resilient Back-propagation* (RPROP) [72] algorithm. This algorithm was chosen because it features a faster convergence rate and produces a better result than the conventional *Backpropagation* [104; 102].

The training set was divided into two parts: 80% for training, and 20% for validation. In addition, the division was performed maintaining the distribution of each class, so the MLP network is capable of estimating the Bayesian *a posteriori probability* [128]. Consequently, their results can be combined through a probabilistic framework.

In every experiment, the number of nodes in the hidden layer was selected by means of the *crossvalidation* method using the training data. The search was performed by varying the number of nodes from 150 to 600 at 10-point intervals. Then, we replicated the configuration that achieved the best results 10 times to obtain the average result. The weights of the neural networks were randomly initialized before each execution.

4.4 Results for the C-CUBE Database

For each feature set, the global information provided by the database (width/height ratio, distance between the baseline and the upper line, distance from the baseline to the upper extremity, and distance from the baseline to the lower extremity) were included in the feature vector. These features contributed to an average increase of two percentile points in the recognition rate.

Two different experiments were performed. The first was conducted to evaluate the performance of the technique for the uppercase and lowercase letters separately (split case). It is important to do this, since some applications need to recognize either uppercase or lowercase letters specifically. The second experiment was conducted using both; however, characters that present similar shapes in the two cases were combined in a single class (joint case). An analysis to verify whether or not the uppercase and lowercase forms of the same letters are similar in shape was performed in [122]. The letters *(C, X, O, W, Y, Z, M, K, J, U, N, F, V)* presented the greatest similarity between the two cases and were combined in a single class. This resulted in 39 classes in the second experiment.

The results for the split and joint cases are shown in Tables III-1 and III-2 respectively. The results are ordered by the recognition rates. The proposed Modified Edge Maps algorithm presented the best result overall.

Table-A III-1 Recognition Rate by Feature Set for the C-Cube database. Uppercase and lowercase letters. # Nodes is the number of nodes in the hidden layer, and Mean is the average performance considering both uppercase and lowercase letters.

Method	# Nodes	Upper Case(%)	Lower Case(%)	Mean(%)
Modified Edge Maps	490	**86.52**	81.13	**83.55** $\pm$ 0.27
Binary Grad.	490	86.35	79.89	82.58 $\pm$ 0.18
MAT Grad.	300	85.77	79.22	81.95 $\pm$ 0.19
Median Grad.	360	85.10	79.48	81.81 $\pm$ 0.21
Camastra 34D	400	79.63	**84.37**	81.74 $\pm$ 0.35
Zoning	450	84.46	78.07	80.74 $\pm$ 0.41
Structural	320	81.94	77.70	79.53 $\pm$ 0.56
Concavities	530	73.35	81.89	76.90 $\pm$ 0.16
Projections	500	71.73	79.90	75.10 $\pm$ 0.39

Most feature sets presented better accuracy for the upper case letters. The exceptions are Image Projections, Concavity Measurement, and Camastra 34D. This fact supports the claim that it is difficult to design a feature extraction method that can deal with the variability of the patterns. In addition, the aim is to recognize both uppercase and lowercase letters, and so it is an advantage to combine techniques that are expert in each task.

4.5 Results for the MNIST database

For the Modified Edge Maps and Directional Gradient methods, the number of nodes in the hidden layer is 300. For the Zoning, Structural Characteristics, Concavity Measurement, and Image Projection techniques, the number of nodes in the hidden layer is 360, 340, 175, and 330 respectively. Table III-3 shows the results for each feature set.

Table III-3 shows that some feature sets have better discriminative power for certain classes of digits. A clear example of this occurs in digits with complex shapes, such as 8 and 9, where

Table-A III-2 Feature set results for the C-Cube database (Joint Case). # Nodes is the number of nodes in the hidden layer.

Method	# nodes	Recognition Rate(%)
Modified Edge Maps	490	**82.49 ± 0.27**
Binary Grad.	490	81.46 ± 0.18
MAT Grad.	300	80.83 ± 0.19
Median Grad.	360	79.96 ± 0.21
Camastra 34D	400	79.97 ± 0.35
Zoning	450	78.60 ± 0.41
Structural	320	77.07 ± 0.56
Concavities	530	74.90 ± 0.16
Projections	500	73.85 ± 0.39

Table-A III-3 Results for each feature extraction method for the MNIST database.

Digit	Structural	Edge Maps	Projections	Multi-Zoning	Concavity	MAT Grad.	Binary Grad.	Median Grad.	Camastra 34D
0	**98.88**	97.86	98.17	**98.88**	96.13	97.96	98.46	98.06	98.46
1	**99.12**	98.15	98.42	98.95	98.33	98.68	99.03	99.11	99.03
2	96.03	95.26	95.26	**96.23**	95.66	95.16	96.22	96.31	96.22
3	96.14	94.76	94.76	**96.84**	91.69	94.46	96.23	96.23	96.23
4	**97.25**	92.15	96.33	97.05	92.98	96.94	98.16	97.45	98.16
5	95.63	94.73	93.61	**96.96**	95.56	96.30	95.62	95.96	95.62
6	**97.81**	96.66	97.18	97.08	96.35	97.39	96.45	96.76	96.45
7	**96.89**	93.77	95.43	95.62	94.38	95.04	95.81	94.94	95.81
8	**96.00**	93.54	93.74	95.90	89.64	95.54	93.83	93.42	93.83
9	**95.60**	90.58	93.85	95.16	92.11	92.66	94.44	95.14	94.44
Mean	**96.95 ± 0.29**	94.78 ± 0.15	95.72 ± 0.13	96.84 ± 0.18	94.31 ± 0.25	95.83 ± 0.13	96.47 ± 0.12	96.38 ± 0.12	96.47 ± 0.32

the difference between the largest and smallest values can be more than six percentile points. For the digits 0, 1, 4, 6, 7, 8, and 9, the Structural Characteristics method achieved the best results, while for the digits 2, 3, and 5, the proposed Multi-Zoning technique obtained a better recognition rate.

The techniques that presented the best results for the MNIST database, Structural Characteristics and Multi-Zoning, are among the worst performers for the C-Cube database (Tables III-1 and III-2). The proposed Modified Edge Maps presented the best accuracy for the C-Cube database, and the second worst result for the MNIST database. This is another reason to use multiple feature extraction techniques.

4.6 Sensitivity Analysis

The validation dataset was used to compute the dissimilarity matrix D and its projection onto the two-dimensional CPS $\tilde{D}$. Figure III-10 shows the CPS for the C-Cube database. Based on visual analysis, four groups of feature representation can be observed. The Modified Edge Maps and Image Projection techniques are a long way from every other point, and can be considered an atomic cluster. The Structural Characteristics and Concavity Measurement techniques make up another group. The last cluster is composed of the gradient methods (MAT Gradient, Binary Gradient, and Median Gradient), as well as the Camastra 34D and Zoning techniques. The fact that the three gradient methods are close to one another is an interesting finding, and the reason for their proximity is that the gradient-based techniques extract similar information (directional), with a slight difference in the preprocessing of the image. The Camastra 34D method also computes directional information.

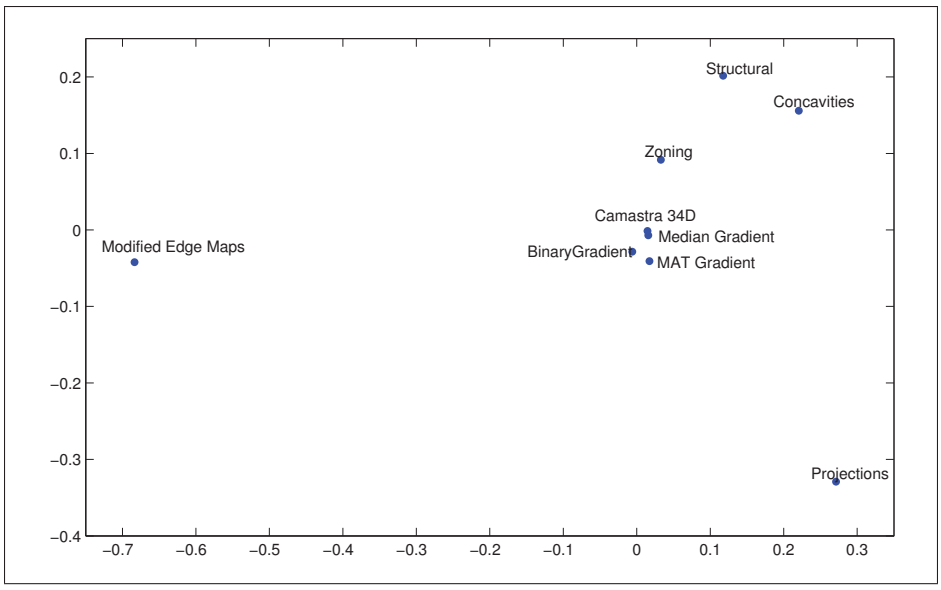


Figure-A III-10 Classifier Projection Space for the C-Cube dataset. The axes of the CPS plot have no significance, and only the distances between the points are important.

Figure III-11 presents the Oracle error analysis for the C-Cube database. We compare feature representations that are next to one another against representations that are far apart. Figure III-

11a shows the Oracle error analysis for three techniques that are close together in the CPS and can be considered to be from the same group. In this case, the three methods that extract information based on gradients (MAT-based Gradient, Binary Gradient, and Median Gradient) are compared. The total number of errors committed using the MAT-based Gradient representation is 3668. Approximately 20% of the errors (692 images) are misclassified by this feature representation alone. The intersection between the MAT-based Gradient and the Binary Gradient methods shows that 2113 images are misclassified, while 1849 images are misclassified when MAT-based and Median Gradient representations are used. In addition, 986 images are misclassified based on the intersection of the three techniques. Therefore, as the majority of errors of these three techniques occur in the same images, combining them is unlikely to result in improved performance.

In contrast, Figure III-11b shows the Oracle error analysis for the MAT-based Gradient with two representations that are far apart in the CPS: Projections, and Edge-Maps. In this case, we can easily see that the majority of errors committed by the MAT-based Gradient, 2058 happens only individually. Both the pair-wise intersections and the intersection of the three techniques produce a much lower number of errors, and only 252 images are misclassified considering these three feature representations. This number is approximately 10 times less than the number of images that are misclassified when only the MAT-based Gradient is considered (2058). So, the errors made by the three techniques occurred in distinct patterns, and therefore can be considered complementary, since they are able to correctly classify different images.

Figure III-12 shows the CPS plot for the MNIST database. We can identify three feature representations that are far away from all the others: Concavities, Zoning, and Structural Characteristics. As with the C-Cube experiment, the results of the gradient-based methods and the Camastra 34D representation are close together, forming a group of similar feature representations. The Modified Edge Maps representation results lie between the Zoning and Projections representation results, and these methods can also be considered to belong to a distinct group.

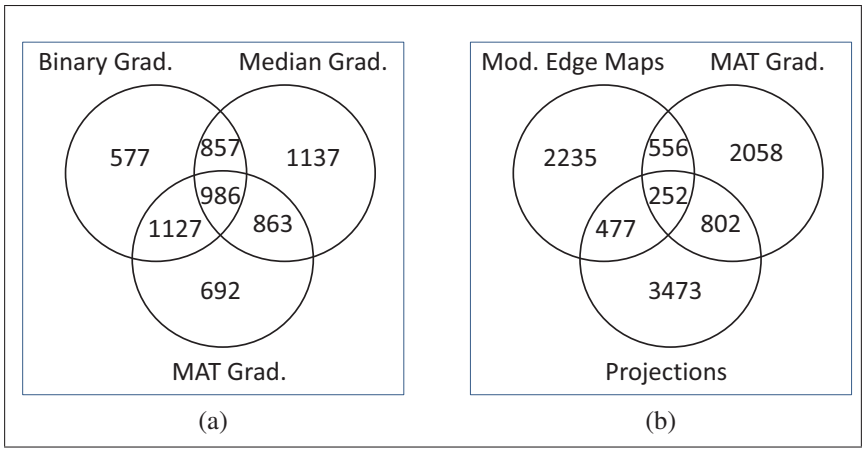


Figure-A III-11 Oracle error analysis for the C-CUBE dataset. (a) Comparison of feature representations that belong to the same cluster. (b) Comparison of feature representations that are far apart.

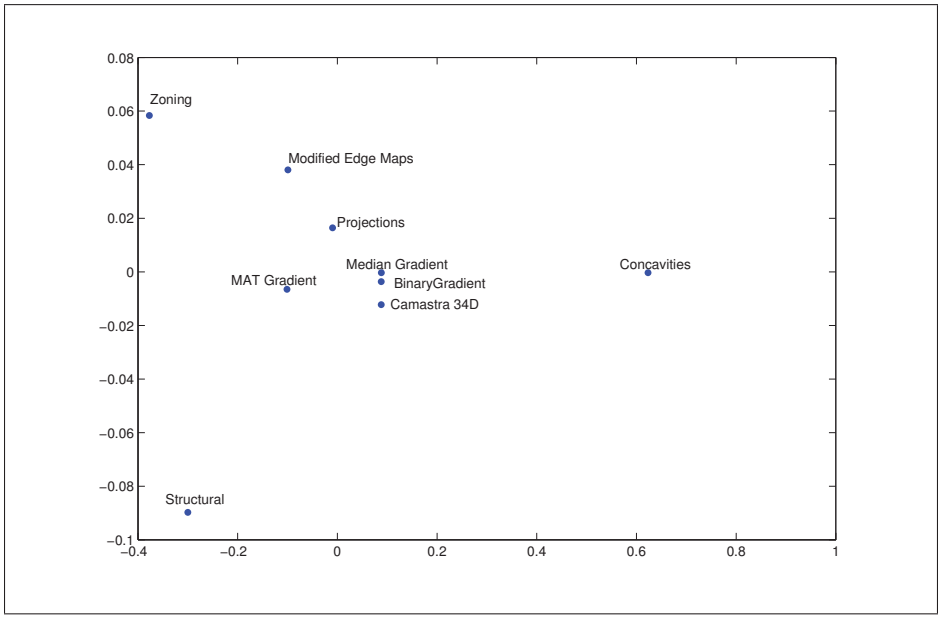


Figure-A III-12 The Classifier Projection Space of the MNIST dataset.

Figure III-13a shows the Oracle error analysis among three methods: Structural Characteristics, Multi-Zoning, and Concavity Measurement. Only nine images were misclassified by the three methods used simultaneously. Moreover, the pairwise intersection of the three techniques also reduces the number of errors. As a result, these three techniques together are able to correctly classify different images. Figure III-13b shows the intersection of three techniques that are

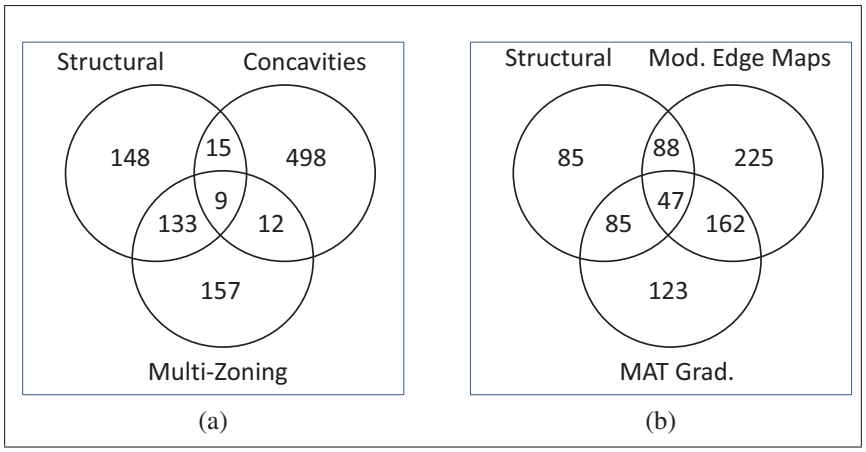


Figure-A III-13 Oracle error analysis for the MNIST dataset. (a) comparison of feature representations that are far apart. (b) comparison of feature representations that form a cloud of points.

closer together on the CPS plot (Structural Characteristics, Edge Maps, and MAT-based Gradient). In this case, the intersection of errors shows that it is possible to reduce the individual errors, since they present complementary information.

So, based on the proposed framework, we can answer two of the questions posed in this paper: Do different feature extraction techniques present complementary information? We demonstrate that different feature extraction techniques are indeed complementary. The majority of the techniques are far apart in the CPS for both datasets. Furthermore, combining them using the Oracle analysis can reduce the individual error by a factor of as high as 10 for the C-Cube dataset (Figure III-11b), and can result in a very low error rate for the MNIST dataset (Figure III-11a).

The exceptions are the representations that extract the gradients of the images. Therefore, in answer to the second question: Are feature extraction techniques that use a similar approach (e.g. different methods to extract gradients) less complementary than techniques that use different characteristics (e.g. edges, concavities)? According to Figures III-10 and III-12, the gradient-based methods based (MAT-based Gradient, Binary Gradient, and Median Gradient) are really close to each other, creating a cloud of points. In addition, the results of the Oracle analysis (Figure III-11(b)) demonstrate that the number of errors that are common to the three

techniques is higher than the number of individual errors. From this, we can conclude that techniques using similar approaches are less complementary, and are likely to misclassify the same images.

5. Multiple-Classifier Systems

MCS have been widely studied as an alternative means of increasing efficiency and accuracy in pattern recognition systems [24; 9; 129]. The main motivation for using classifier ensembles comes from the observation that errors committed by classifiers trained with different feature extraction methods do not overlap. Another reason to use them is based on the divide-and-conquer paradigm: instead of using a single set consisting of all feature sets, the idea is to use each feature extraction method separately and combine their results. There are many examples in the literature that show the efficiency of an ensemble of classifiers in various tasks, such as signature verification [101], pedestrian detection [130], and image labeling [28].

The advantage of combining classifiers that deal with distinct feature sets is that they represent different transformations of the image into the feature space. Suppose, for example, that a pattern is located near the decision boundary. The recognition of this pattern is a difficult task in the feature space used. It is still difficult when multiple classifiers are applied over the same feature space. However, if different feature spaces are used, this pattern might be close to the decision boundary in one feature space, but the same pattern might be far from the decision boundary of another feature space, as its transformation is completely different. In this way, the pattern can be easily recognized.

5.1 Trained Combiner

Duin [7] concluded that fixed combination rules only achieve the best results in very strict conditions. Normally, these results are suboptimal, and the performance of these rules falls far short of the performance of the Oracle. For instance, the Product rule is known to fail if one of the classifiers' estimates is close to zero, or is accidentally zero. So, if one feature set is not suitable for the query image, the system is likely to fail. The majority vote rule only produces

the correct classification if at least half the classifiers predict the correct class. However, there are certain images that are correctly classified in only one or two of the nine feature sets, and so we cannot achieve a performance close to the Oracle using this combination rule either.

Consequently, we decided to use a trained combiner in order to achieve a more robust combination of classifiers. Trained combiners usually perform better, since the combiner can adapt to the classification problem [7]. In this methodology, the outputs of the base classifiers are used as input features for a new classifier that is trained to aggregate the results. During the training phase, the combiner learns how to deal with difficult situations, such as, for example, when a small subset of the base classifiers produces the correct answer.

In the experimental study, the trained combiner is an MLP network with one hidden layer. Neural networks are good candidates for use as trained combiners, because they are robust to noise. This means that the MLP combiner can still predict the correct output, even when the majority of the base classifiers present errors.

5.2 Experimental Protocol

In this section, the results obtained by combining the feature extraction techniques are presented. For the combination module, the MLP combiner is compared to well-known fixed combination rules. The fixed rules considered are Sum, Product, Maximum, Median, Voting, and Oracle. The theoretical framework for the fixed combination rules is described in [24; 119].

The experiment was conducted using 10 iterations, in order to obtain the mean and standard deviation for the results. For each iteration, the base classifiers were retrained following the protocol described in Section 4.3. This replication is important, since the results are sensitive to the initial weight configuration of the base classifiers.

For each image in the training set, the *a posteriori probability* for each feature set is estimated and used as an input feature to train the MLP combiner. Two experiments were conducted using this combiner: MLP_{all}, which consists of the nine feature representations, and $\text{MLP}_{\text{selection}}$,

which consists of a subset using feature representations selected based on the sensitivity analysis.

For the $MLP_{selection}$ configuration, the MAT-based Gradient, Binary Gradient, Median Gradient, and Camastra 34D techniques are considered redundant for both datasets (Section 4.6). As we use only the Binary Gradient to represent this group of techniques, because it achieved the highest accuracy, the configuration $MLP_{selection}$ consists of only 6 feature representations: Modified Edge Maps, Concavity Measurement, Multi-Zoning, Structural Characteristics, Binary Gradient, and Image Projections.

In every experiment, combiner training is accomplished using the *Resilient Backpropagation* algorithm. The number of nodes in the hidden layer of the MLP combiner was selected using the *crossvalidation* method with the training data. The search was conducted by varying the number of nodes from 10 to 300 at 10-point intervals. The number of nodes in the hidden layer of the MLP combiner for the C-Cube and MNIST datasets were 300 and 50 respectively.

5.3 Results for the C-Cube Dataset

Tables III-4 and III-5 show the results of the combination for the C-Cube database. A Kruskal-Wallis non parametric statistical test (95% confidence level) applied to the difference in accuracy rates showed that the results with the combination rules are statistically significant when compared to the classifiers trained using a single feature extraction technique. This can be explained by the fact that the feature extraction techniques considered in this analysis present complementary information; the majority of them are far apart in the CPS (Figure III-10). This means that the recognition performance could be improved any combination rule.

The only exception was the Product rule. Its results for the separate case were not statistically better than those of the Modified Edge Maps technique. This might be explained by the fact that there was a large difference in the accuracy of the feature representations.

Table-A III-4 Results of each combination method for the C-Cube database. Uppercase and lowercase letters.

Method	Upper Case(%)	Lower Case(%)	Mean(%)
Sum	91.21	86.94	88.92
Product	85.92	79.52	82.37
Maximum	89.83	85.22	87.14
Median	91.00	87.33	88.86
Maj. Vote	90.99	87.44	88.92
MLP_{all}	**91.39**	**88.45**	**89.67**
$MLP_{selection}$	**90.89**	**88.25**	**88.85**
Oracle	96.87	97.24	97.09

Table-A III-5 Results of each combination rule for the C-Cube database (Joint case).

Method	Best (%)	Mean (%)
Sum	88.51	88.22 ± 0.19
Product	86.99	85.52 ± 0.89
Maximum	85.48	85.73 ± 0.67
Median	88.84	88.04 ± 0.53
Maj. Vote	89.22	88.00 ± 0.81
MLP_{all}	**89.65**	$\mathbf{89.28 \pm 0.22}$
$MLP_{selection}$	**89.54**	$\mathbf{88.98 \pm 0.50}$
Oracle	97.78	97.25 ± 1.72

Figure III-14 shows the box plot with the results for the combination rules for the C-Cube database. The gain in recognition performance for the MLP combiner is statistically significant when compared with that of the fixed combination rules. The MLP_{all} combiner presented the best mean result. However, based on the Kruskal-Wallis test, the results were not statistically better than those of the reduced combination, $MLP_{selection}$.

5.4 Results for the MNIST Database

Table III-6 shows the results obtained by the combination methods for the MNIST database. The recognition performance of all the combination rules was a great improvement over all the (feature extraction, classifier) pairs shown in Table III-3. The Kruskal-Wallis non parametric

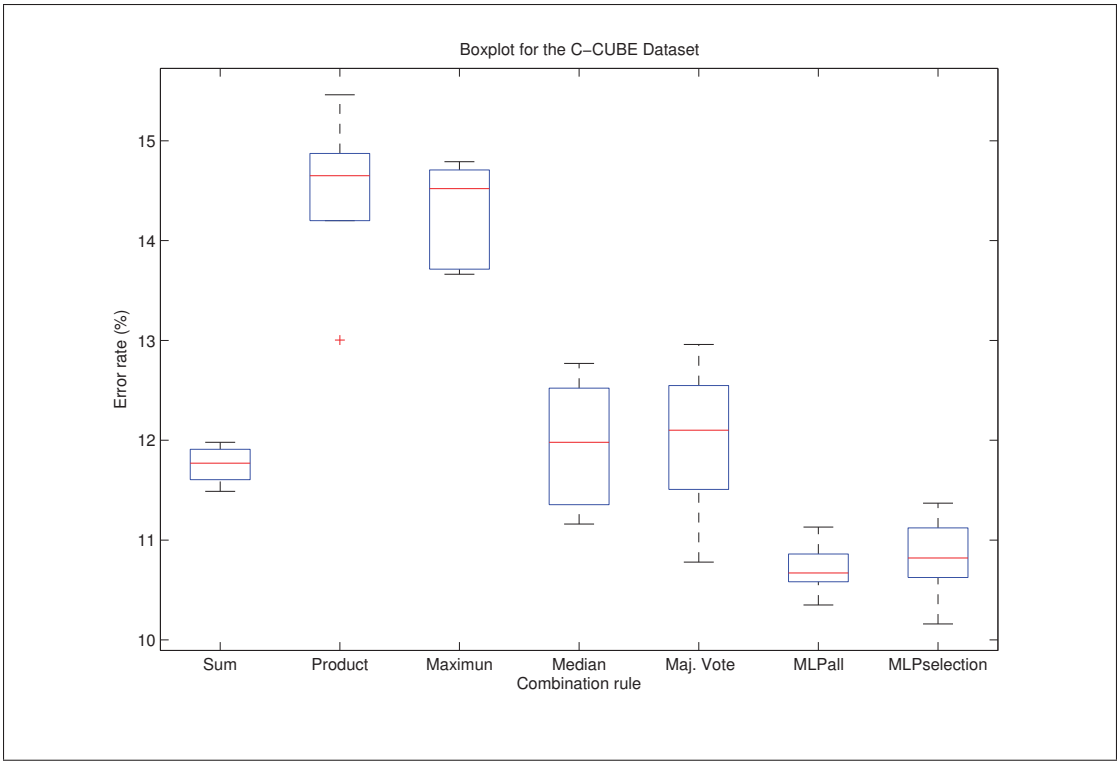


Figure-A III-14 Boxplot diagram comparing the combination rules for the CCUBE database. MLP_{all} and $MLP_{selection}$ are the MLP combiner for the experiment using every feature representation and the reduced feature representation set respectively.

statistical test with a 95% confidence level was also used, and the result obtained by every combination rule was statistically better. Once again, the gain in performance is explained by the fact that the majority of feature representations considered presents complementary information.

As with the C-Cube dataset, the trained combiner outperformed the other combination rules. The $MLP_{selection}$ combination achieved an accuracy rate close to the Oracle performance (which was 100%). This is due to the ability of the network to learn how to perform the best combination using the training set. In addition, the standard deviation of the trained combiner is 0.04%, which is approximately six times less than the standard deviation for the Maximum rule. Even when one or more feature sets produce a very inaccurate result, the trained combiner is still able to predict the correct output. Figure III-15 shows the box plot for the combination rules.

Table-A III-6 Results of each combination rule for the MNIST database.

Method	Best (%)	Mean ± std dev (%)
Sum	99.23	98.96 ± 0.42
Product	99.55	99.27 ± 0.34
Maximum	99.58	99.43 ± 0.23
Median	99.12	98.85 ± 0.25
Maj. Vote	98.98	98.63 ± 0.49
MLP_{all}	**99.72**	**99.70 ± 0.01**
$MLP_{selection}$	**99.76**	**99.72 ± 0.04**
Oracle	100	100 ± 0.00

The median result of both MLP_{all} and $MLP_{selection}$ achieved a lower error rate than the best results of the other combination rules.

Furthermore, the result of the MLP combiner is followed by the Maximum rule that also presented a high recognition rate. This is because of the ability that some of the feature extraction methods have to recognize certain types of digits.

In both experiments, the results using all the feature representations (MLP_{all}) and the configuration following the sensitivity analysis ($MLP_{selection}$) are statistically equivalent. Nevertheless, for the C-Cube dataset, the $MLP_{selection}$ achieved a result 0.04 percentile points higher than that of MLP_{all}. This is an interesting finding, since $MLP_{selection}$ is composed of a small number of feature representations. The redundant nature of MLP_{all} might interfere with the performance of the combination. This means that we can answer the third question posed in the introduction, as follows: The proposed framework selects feature representations that can be used to construct an efficient MCS in terms of accuracy rates.

5.5 Computational Time

Analyzing the proposed system when the trained combiner is used, the average computational time per image is 4 milliseconds for the MNIST and 9 milliseconds for the C-Cube dataset. The application was developed using C++ running on a 2.40 Ghz machine with four cores.

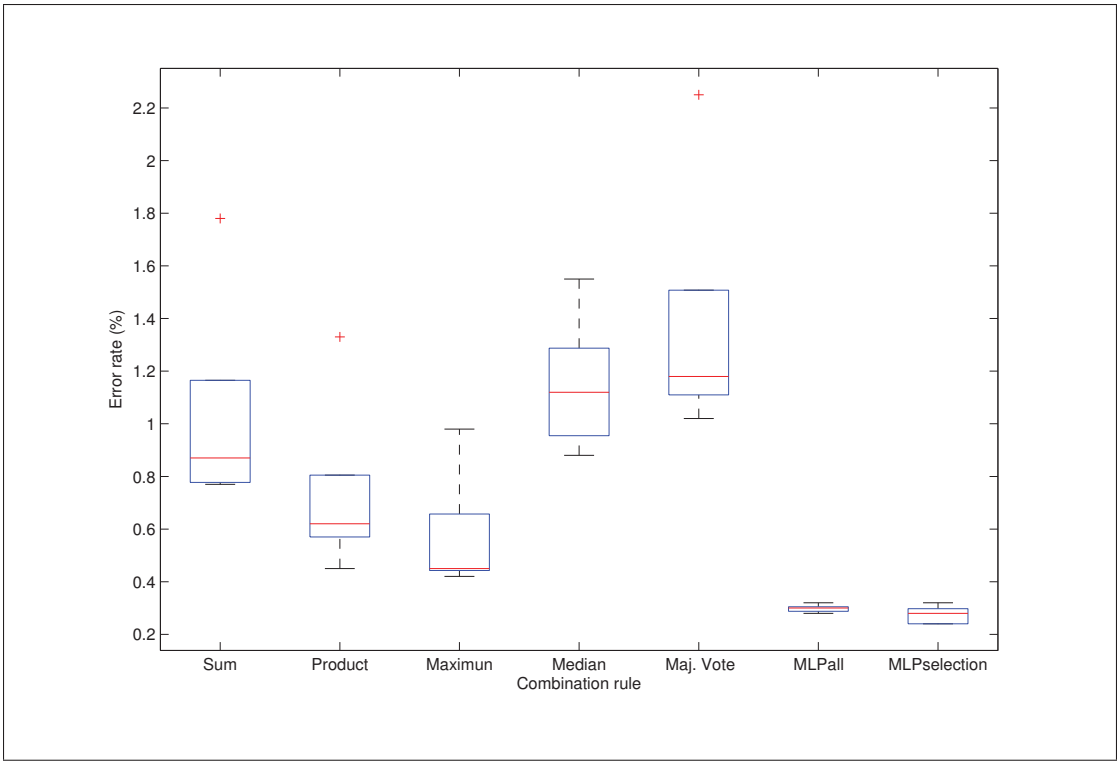


Figure-A III-15 Boxplot diagram comparing the combination rules for the MNIST database. MLP_{all} and $MLP_{selection}$ are the MLP combiner for the experiment using every feature representation and the reduced feature representation set respectively.

We measured the difference in computational cost of the MLP combiner and the fixed combination rule. That difference is measured in microseconds, and does not affect the overall computational time of the system. This was expected, since the MLP combiner has a total of 30,000 connections (60 inputs $\times$ 50 hidden nodes $\times$ 10 output nodes), while the network trained with the Structural Characteristics feature set has 952,000 connections (280 inputs $\times$ 340 hidden nodes $\times$ 10 output nodes). In other words, the cost of computing the combination is approximately 31 times less than the cost of computing a single feature set.

5.6 Comparison with the State of the Art

The best results obtained for the C-Cube database are shown in Table III-7. To the best of our knowledge, the proposed combination scheme outperforms all the previous results in the *Split B* of this database. Furthermore, it is important to observe that the past best results are based on

Support Vector Machines (SVM) using the one-versus-the-rest approach [131]. This method trains one specific classifier for each class. For this problem, a large number of classifiers is required, which is one of the drawbacks of these approaches. As far as we know, the proposed system is the first to show high accuracy using only MLPs.

Table-A III-7 Comparative results for the C-Cube database. RBF = Radial Basis Network with 5120 centers, HVQ = Hierarchical Vector Quantization, MDF = Modified Directional Features, SVM = SVM with Radial basis Kernel.

Algorithm	**Recognition Rate(%)**
HVQ-32 [132]	84.72
HVQ-16 [132]	85.58
MDF-RBF [127]	80.92
34D-RBF [127]	84.27
MDF-SVM [127]	83.60
34D-SVM + Neural GAS [122]	86.20
34D-MLP [122]	71.42
Proposed	**89.28 $\pm$ 0.22**

The best results obtained for the MNIST database are shown in Table III-8. The proposed combination scheme outperformed all the previous results for this database. It is also important to observe that many of the best results [133; 134; 135; 136; 137; 138] are based on large neural networks, such as Convolutional Neural Networks or Deep Neural Networks. In addition, the techniques used previously need to expand the training data by creating new images through distortions [133; 134; 135; 136; 139; 138]. Our approach to achieving high performance in handwritten recognition is different, in that no additional training data is required.

6. Conclusion

We have proposed a new framework for analyzing the relationship between different feature representations. Each representation is used to train a single classifier, and the dissimilarities between them are computed to generate a dissimilarity matrix. Through the Multidimensional Scaling method (Sammon Mapping), this dissimilarity matrix is embedded in a two-

Table-A III-8 Comparative Results for the MNIST Database.

Method	**Distortions**	**Recognition Rate(%)**
Boosted LeNet-4 [133]	Affine	99.30
unsupervised sparse features + SVM [140]	-	99.41
Trainable feature extractor + SVM [134]	Affine	99.46
Large convolutional network + unsupervised pretraining [137]	-	99.47
PNCN classifier [139]	Skewing	99.56
Cascade ensemble classifier (without rejection) [141]	-	99.59
Convolutional neural networks [135]	Elastic	99.60
Large convolutional network + unsupervised pretraining [136]	Elastic	99.61
6 Layers MLP 841-2500-2000-1500-1000-500-10 [138]	Elastic	99.65
Proposed	**-**	**99.72 $\pm$ 0.04**

dimensional space (CPS) where the Euclidean distance between two feature representations reflects their dissimilarity. Based on this two-dimensional plot, a sensitivity analysis is performed in order to determine whether the representations are complementary or redundant.

We have applied the proposed framework to two handwritten recognition datasets: the Cursive Character Challenge (C-Cube) for handwritten letters, and the MNIST dataset for handwritten digits. The results demonstrate that feature representations using distinct approaches (edges, projections, gradient, and concavities) extract information that is dissimilar. Consequently, they are complementary. Techniques that use the same observations, using a different rule to compute the features (e.g. the MAT-based Gradient, Median Gradient, and Binary Gradient) perform in a similar fashion. They appear close to each other in both experiments and are likely to commit errors on the same images. As a result, they can be considered redundant.

A multiple-classifier system using distinct feature extraction techniques was designed based on the feature representation analysis. As the majority of techniques considered present complementary information, the results of every combination rule outperform the best individual classifier for both datasets. With the aim of searching for the optimal combination rule, we used a neural network as a combiner. The results show that the proposed approach presents better accuracy when compared with state-of-the-art techniques.

The two experiments that were performed: one using all the feature representations, and the other a reduced set of representations based on a sensitivity analysis, demonstrate that the strategies are statistically equivalent. In some cases, the reduced set of representations can even achieve higher performance, as redundant classifiers can negatively affect performance. This shows that our framework can also be used to perform feature representation selection. In this paper, however, we use the empirical analysis of the CPS and the Oracle error analysis manually, in order to make this selection. An algorithm designed to perform the selection automatically using our framework is currently being developed.

APPENDIX IV

ANALYZING DYNAMIC ENSEMBLE SELECTION TECHNIQUES USING DISSIMILARITY ANALYSIS

Rafael M. O. Cruz[1], Robert Sabourin[1], George D. C. Cavalcanti[2]

[1] Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), École de Technologie Supérieure (ÉTS), Montréal, Canada H3C 1K3

[2] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brazil

Article Published in « Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR) » 2014.

Abstract

In Dynamic Ensemble Selection (DES), only the most competent classifiers are selected to classify a given query sample. A crucial issue faced in DES is the definition of a criterion for measuring the level of competence of each base classifier. To that end, a criterion commonly used is the estimation of the competence of a base classifier using its local accuracy in small regions of the feature space surrounding the query instance. However, such a criterion cannot achieve results close to the performance of the Oracle, which is the upper limit performance of any DES technique. In this paper, we conduct a dissimilarity analysis between various DES techniques in order to better understand the relationship between them and as well as the behavior of the Oracle. In our experimental study, we evaluate seven DES techniques and the Oracle using eleven public datasets. One of the seven DES techniques was proposed by the authors and uses meta-learning to define the competence of base classifiers based on different criteria. In the dissimilarity analysis, this proposed technique appears closer to the Oracle when compared to others, which would seem to indicate that using different bits of information on the behavior of base classifiers is important for improving the precision of DES techniques. Furthermore, DES techniques, such as LCA, OLA, and MLA, which use similar criteria to define the level of competence of base classifiers, are more likely to produce similar results.

1. Introduction

In recent years, ensembles of Classifiers (EoC) have been widely studied as an alternative for increasing efficiency and accuracy in pattern recognition [24; 9]. Classifier ensembles involve two basic approaches, namely, classifier fusion and dynamic ensemble selection. With classifier fusion approaches, each classifier in the ensemble is used, and their outputs are aggregated to give the final prediction. However, such techniques [24; 104] present two main problems: they are based on the assumption that the base classifiers commit independent errors, which rarely occurs to find in real pattern recognition applications.

On the other hand, Dynamic Ensemble Selection (DES) techniques [1] rely on the assumption that each base classifier[1] is an expert in a different local region of the feature space. DES techniques work by measuring the level of competence of each base classifier, considering each new test sample. Only the most competent(s) classifier(s) is(are) selected to predict the class of a new test sample. Hence, the key issue in DES is defining a criterion for measuring the level of competence of a base classifier. Most DES techniques [14; 22; 20; 29] use estimates of the classifier's local accuracy in small regions of the feature space surrounding the query instance as search criteria to carry out the ensemble selection. However, in our previous work [20], we demonstrated that this criterion is limited, and cannot achieve results close to the performance of the Oracle, which represents the best possible result of any combination of classifiers [9]. In addition, as reported by Ko et al. [14], addressing the behavior of the Oracle is much more complex than applying a simple neighborhood approach, and the task of figuring out its behavior based merely on the pattern feature space is not an easy one.

To tackle this issue, in [36] we proposed a novel DES framework in which multiple criteria regarding the behavior of a base classifier are used to compute its level of competence. In this paper, we conduct a dissimilarity analysis between different DES techniques in order to better understand their relationship. The analysis is performed based on the difference between the levels of competence of a base classifier estimated by the criterion embedded in each DES

[1]The term base classifier refers to a single classifier belonging to an ensemble or a pool of classifiers

technique. All in all, we compare the DES criteria of seven state-of-the-art DES techniques, including our proposed meta-learning framework. In addition, we also formalize the Oracle as an ideal DES technique (i.e., a DES scheme which selects only the classifiers of the pool that predict the correct class for the query instance) to be used in the analysis.

The dissimilarities between different DES criteria are computed in order to generate a dissimilarity matrix, which is then, used to project each DES technique onto a two-dimensional space, called the Classifier Projection Space (CPS) [113]). In the CPS, each DES technique is represented by a point, and the distance between two points corresponds to their degree of dissimilarity. Techniques that appear close together present similar behavior (i.e., they are more likely to produce the same results), while those appearing far apart in the two-dimensional CPS can be considered different. Thus, a spatial relationship is achieved between different techniques. The purpose of the dissimilarity analysis is twofold: to understand the relationship between different DES techniques (i.e., whether or not the criteria used by DES techniques present a similar behavior), and in order to determine which DES technique presents a behavior that is closer to the behavior of the ideal DES scheme (Oracle).

This paper is organized as follows: Section 2 introduces the DES techniques from the literature that are used in the dissimilarity analysis. The proposed meta-learning framework is described in Section 3. Experiments are conducted in Section 4, and finally, our conclusion is presented in the last section.

2. Dynamic ensemble selection techniques

The goal of dynamic selection is to find an ensemble of classifiers, $C' \subset C$ containing the best classifiers to classify a given test sample $\mathbf{x}_j$. This is different from static selection, where the ensemble of classifiers C' is selected during the training phase, and considering the global performance of the base classifiers over a validation dataset. In dynamic selection, the classifier competence is measured on-the-fly for each query instance $\mathbf{x}_j$.

The following DES techniques are described in this section: Overall Local Accuracy (OLA) [22], Local Classifier Accuracy (LCA) [22], Modified Local Accuracy (MLA) [29], KNORA-Eliminate [14], K-Nearest Output Profiles (KNOP) [16] and Multiple Classifier Behavior (MCB) [21].

For the definitions below, let $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ be the region of competence of the test sample $\mathbf{x}_j$ (K is the size of the region of competence), defined on the validation data, c_i a base classifier from the pool $C = \{c_1, \ldots, c_M\}$ (M is the size of the pool), w_l the correct label of $\mathbf{x}_j$ and $\delta_{i,j}$ the level of competence of c_i for the classification of the input instance $\mathbf{x}_j$.

Overall Local Accuracy (OLA)

In this method, the level of competence $\delta_{i,j}$ of a base classifier c_i is simply computed as the local accuracy achieved by c_i for the region of competence θ_j. (Equation A IV-1). The classifier with the highest level of competence $\delta_{i,j}$ is selected.

$$\delta_{i,j} = \sum_{k=1}^{K} P(w_l \mid \mathbf{x}_k \in w_l, c_i) \tag{A IV-1}$$

Local Classifier Accuracy (LCA)

This rule is similar to the OLA, with the only difference being that the local accuracy of c_i is estimated with respect to the output classes; w_l (w_l is the class assigned for $\mathbf{x}_j$ by c_i) for the whole region of competence, θ_j (Equation A IV-2). The classifier with the highest level of competence $\delta_{i,j}$ is selected.

$$\delta_{i,j} = \frac{\sum_{\mathbf{x}_k \in w_l} P(w_l \mid \mathbf{x}_k, c_i)}{\sum_{k=1}^{K} P(w_l \mid \mathbf{x}_k, c_i)} \tag{A IV-2}$$

Modified Local Accuracy (MLA)

The MLA technique works similarly to the LCA. The only difference is that each instance $\mathbf{x}_k$ belonging to the region of competence θ_j is weighted by its Euclidean distance to the query sample $\mathbf{x}_j$. The classifier with the highest level of competence $\delta_{i,j}$ is selected.

KNORA-Eliminate (KNORA-E)

Given the region of competence θ_j, only the classifiers that achieved a perfect score, considering the whole region of competence, are considered competent for the classification of $\mathbf{x}_j$. Thus, the level of competence $\delta_{i,j}$ is either "competent", $\delta_{i,j} = 1$ or "incompetent", $\delta_{i,j} = 0$. All classifiers considered competent are selected.

Multiple Classifier Behavior (MCB)

Given the query pattern $\mathbf{x}_j$, the first step is to compute its K-Nearest-Neighbors $\mathbf{x}_k, k = 1, \ldots, K$. Then, the output profiles of each neighbor $\tilde{\mathbf{x}}_k$ are computed and compared to the output profile of the test instance $\tilde{\mathbf{x}}_j$ according to a similarity metric $D_{OutProf}$. If $D_{OutProf} > threshold$, the pattern is removed from the region of competence. The level of competence $\delta_{i,j}$ is measured by the recognition performance of the base classifier c_i over the filtered region of competence. The classifier with the highest level of competence $\delta_{i,j}$ is selected.

K-Nearest Output Profiles (KNOP)

This rule is similar to the KNORA technique, with the only difference being that KNORA works in the feature space while KNOP works in the decision space using output profiles. First, the output profiles' transformation is applied over the input $\mathbf{x}_j$, giving $\tilde{\mathbf{x}}_j$. Next, the similarity between $\tilde{\mathbf{x}}_j$ and the output profiles from the validation set is computed and stored in the set ϕ_j. The level of competence $\delta_{i,j}$ of a base classifier c_i for the classification of $\mathbf{x}_j$ is defined by the number of samples in ϕ_j that are correctly classified by c_i.

Oracle

The Oracle is classically defined in the literature as a strategy that correctly classifies each query instance $\mathbf{x}_j$ if any classifier c_i from the pool of classifiers C predicts the correct label for $\mathbf{x}_j$. In this paper, we formalize the Oracle as the ideal DES technique which always selects the classifier that predicts the correct label $\mathbf{x}_j$ and rejects otherwise. The Oracle as a DES technique is defined in Equation A IV-3:

$$\begin{cases} \delta_{i,j} = 1, & \text{if } c_i \text{ correctly classifies } \mathbf{x}_j \\ \delta_{i,j} = 0, & \text{otherwise} \end{cases} \tag{A IV-3}$$

In other words, the level of competence $\delta_{i,j}$ of a base classifier c_i is 1 if it predicts the correct label for $\mathbf{x}_j$, or 0 otherwise.

3. Dynamic ensemble selection using meta-learning

A general overview of the proposed meta-learning framework is depicted in Figure IV-1. It is divided into three phases: Overproduction, Meta-training and Generalization. Each phase is detailed in the following sections.

3.1 Overproduction

In this step, the pool of classifiers $C = \{c_1, \ldots, c_M\}$, where M is the pool size, is generated using the training dataset $\mathcal{T}$. The Bagging technique [3] is used in this work in order to build a diverse pool of classifiers.

3.2 Meta-Training

In this phase, the meta-features are computed and used to train the meta-classifier λ. As shown in Figure IV-1, the meta-training stage consists of three steps: sample selection, the meta-

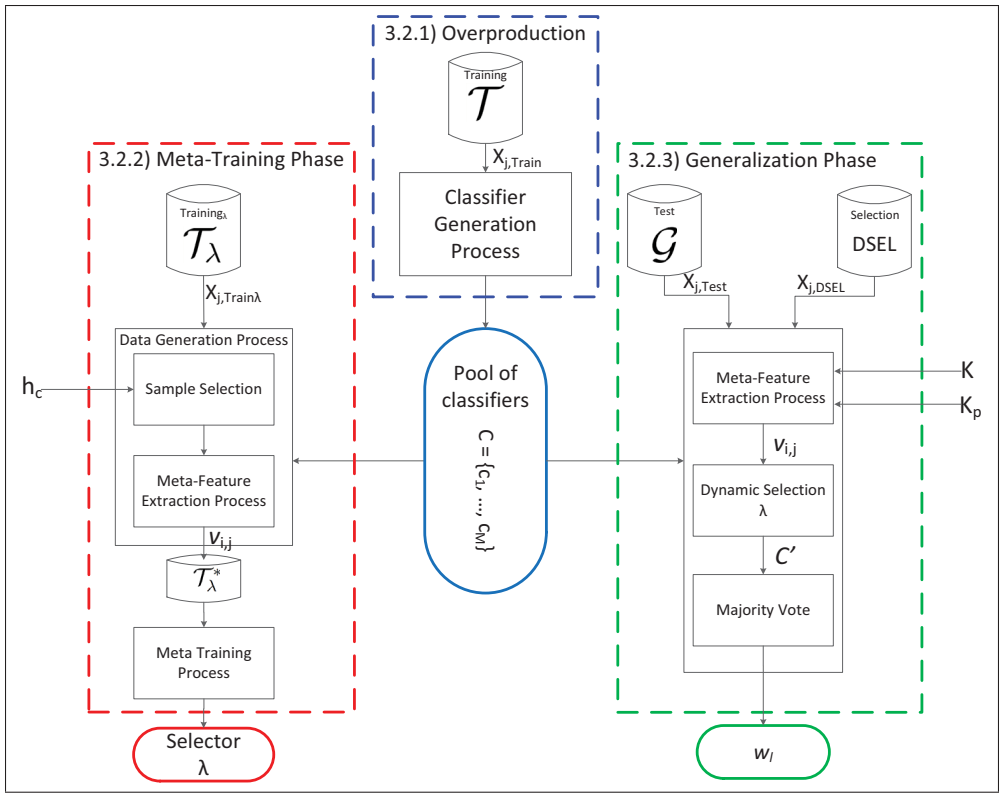


Figure-A IV-1 Overview of the proposed framework. It is divided into three steps 1) Overproduction 2) Meta-training and 3) Generalization [Adapted from [36]]

features extraction process and meta-training. A different dataset $\mathcal{T}_\lambda$ is used in this phase to prevent overfitting.

3.2.1 Sample selection

We focus the training of λ on cases in which the extent of consensus of the pool is low. Thus, we employ a sample selection mechanism based on a threshold h_C, called the consensus threshold. For each $\mathbf{x}_{j,train_\lambda} \in \mathcal{T}_\lambda$, the degree of consensus of the pool, denoted by $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$, is computed. If $H\left(\mathbf{x}_{j,train_\lambda}, C\right)$ falls below the threshold h_C, $\mathbf{x}_{j,train_\lambda}$ is passed down to the meta-features extraction process.

3.2.2 Meta-features extraction

In order to extract the meta-features, the region of competence of $\mathbf{x}_{j,train_\lambda}$, denoted by $\theta_j = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ must be first computed. The region of competence is defined in the $\mathcal{T}_\lambda$ set using the K-Nearest Neighbor algorithm. Then, $\mathbf{x}_j$ is transformed into an output profile, $\tilde{\mathbf{x}}_j$ by applying the transformation T, $(T : \mathbf{x}_j \Rightarrow \tilde{\mathbf{x}}_j)$, where $\mathbf{x}_j \in \Re^D$ and $\tilde{\mathbf{x}}_j \in Z^M$ [16]. The output profile of a pattern $\mathbf{x}_j$ is denoted by $\tilde{\mathbf{x}}_j = \left\{\tilde{\mathbf{x}}_{j,1}, \tilde{\mathbf{x}}_{j,2}, \ldots, \tilde{\mathbf{x}}_{j,M}\right\}$, where each $\tilde{\mathbf{x}}_{j,i}$ is the decision yielded by the classifier c_i for $\mathbf{x}_j$. The similarity between $\tilde{\mathbf{x}}_j$ and the output profiles of the instances in $\mathcal{T}_\lambda$ is obtained through the Euclidean distance. The most similar output profiles are selected to form the set $\phi_j = \left\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{K_p}\right\}$, where each output profile $\tilde{\mathbf{x}}_k$ is associated with a label $w_{l,k}$. Next, for each base classifier $c_i \in C$, five sets of meta-features are calculated:

f_1 **- Neighbors' hard classification:** First, a vector with K elements is created. For each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of the vector is set to 1, otherwise it is 0. Thus, K meta-features are computed.

f_2 **- Posterior probability:** First, a vector with K elements is created. Then, for each instance $\mathbf{x}_k$, belonging to the region of competence θ_j, the posterior probability of c_i, $P(w_l \mid \mathbf{x}_k)$ is computed and inserted into the k-th position of the vector. Consequently, K meta-features are computed.

f_3 **- Overall local accuracy:** The accuracy of c_i over the whole region of competence θ_j is computed and encoded as f_3.

f_4 **- Output profiles classification:** First, a vector with K_p elements is generated. Then, for each member $\tilde{\mathbf{x}}_k$, belonging to the set of output profiles ϕ_j, if the label produced by c_i for $\mathbf{x}_k$ is equal to the label $w_{l,k}$ of $\tilde{\mathbf{x}}_k$, the k-th position of the vector is set to 1, otherwise it is 0. A total of K_p meta-features are extracted using output profiles.

f_5 **- Classifier's Confidence:** The perpendicular distance between the input sample $\mathbf{x}_{j,train_\lambda}$ and the decision boundary of the base classifier c_i is calculated and encoded as f_5. f_5 is normalized to a $[0-1]$ range using the Min-max normalization.

A vector $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ is obtained at the end of the process. It is important to mention that a different vector $v_{i,j}$ is generated for each base classifier c_i. If c_i correctly classifies $\mathbf{x}_{j,train_\lambda}$, the class attribute of $v_{i,j}$, $\alpha_{i,j} = 1$ (i.e., $v_{i,j}$ corresponds to the behavior of a competent classifier), otherwise $\alpha_{i,j} = 0$. $v_{i,j}$ is stored in the meta-features dataset (Figure IV-1).

3.2.3 Training

With the meta-features dataset, $\mathcal{T}_\lambda^*$, on hand, the last step of the meta-training phase is the training of the meta-classifier λ. The dataset $\mathcal{T}_\lambda^*$ is divided on the basis of 75% for training and 25% for validation. A Multi-Layer Perceptron (MLP) neural network with 10 neurons in the hidden layer is considered as the selector λ. The training process for λ is performed using the Levenberg-Marquadt algorithm, and is stopped if its performance on the validation set decreases or fails to improve for five consecutive epochs.

3.3 Generalization

Given an input test sample $\mathbf{x}_{j,test}$ from the generalization dataset $\mathcal{G}$, first, the region of competence θ_j and the set of output profiles ϕ_j, are calculated using the samples from the dynamic selection dataset D_{SEL} (Figure IV-1). For each classifier $c_i \in C$, the five subsets of meta-features are extracted, returning the meta-features vector $v_{i,j}$. Next, $v_{i,j}$ is passed down as input to the meta-classifier λ, which decides whether c_i is competent enough to classify $\mathbf{x}_{j,test}$. In this case, the posterior probability obtained by the meta-classifier λ is considered as the estimation of the level of competence $\delta_{i,j}$ of the base classifier c_i in relation to $\mathbf{x}_{j,test}$.

After each classifier of the pool is evaluated, the majority vote rule [9] is applied over the ensemble C', giving the label w_l of $\mathbf{x}_{j,test}$. Tie-breaking is handled by choosing the class with the highest a posteriori probability.

4. Experiments

We evaluated the generalization performance of the proposed technique using eleven classification datasets, nine from the UCI machine learning repository, and two artificially generated using the Matlab PRTOOLS toolbox[2]. The experiment was conducted using 20 replications. For each replication, the datasets were randomly divided on the basis of 25% for training ($\mathcal{T}$), 25% for meta-training $\mathcal{T}_\lambda$, 25% for the dynamic selection dataset (D_{SEL}) and 25% for generalization ($\mathcal{G}$). The divisions were performed while maintaining the prior probability of each class. The pool of classifiers was composed of 10 Perceptrons. The values of the hyper-parameters K, K_p and h_c were set as 7, 5 and 70%, respectively. They were selected empirically based on previous publications [20; 36].

4.1 Results

Table-A IV-1 Mean and standard deviation results of the accuracy obtained for the proposed meta-learning framework and the DES systems in the literature. The best results are in bold. Results that are significantly better ($p < 0.05$) are underlined

Database	Proposed	KNORA-E	MCB	LCA	OLA	MLA	KNOP	Oracle
Pima	**77.74(2.34)**	73.16(1.86)	73.05(2.21)	72.86(2.98)	73.14(2.56)	73.96(2.31)	73.42(2.11)	95.10(1.19)
Liver Disorders	**68.83 (5.57)**	63.86(3.28)	63.19(2.39)	62.24(4.01)	62.05(3.27)	57.10(3.29)	65.23(2.29)	90.07(2.41)
Breast Cancer	**97.41(1.07)**	96.93(1.10)	96.83(1.35)	97.15(1.58)	96.85(1.32)	96.66(1.34)	95.42(0.89)	99.13(0.52)
Blood Transfusion	**79.14(1.88)**	74.59(2.62)	72.59(3.20)	72.20(2.87)	72.33(2.36)	70.17(3.05)	77.54(2.03)	94.20(2.08)
Banana	**90.16(2.09)**	88.83(1.67)	88.17(3.37)	89.28(1.89)	89.40(2.15)	80.83(6.15)	85.73(10.65)	94.75(2.09)
Vehicle	**82.50(2.07)**	81.19(1.54)	80.20(4.05)	80.33(1.84)	81.50(3.24)	71.15(3.50)	80.09(1.47)	96.80(0.94)
Lithuanian Classes	**90.26(2.78)**	88.83(2.50)	89.17(2.30)	88.10(2.20)	87.95(1.85)	77.67(3.20)	89.33(2.29)	98.35 (0.57)
Sonar	**79.72(1.86)**	74.95(2.79)	75.20(3.35)	76.51(2.06)	74.52(1.54)	74.85(1.34)	75.72(2.82)	94.46(1.63)
Ionosphere	**89.31(0.95)**	87.37(3.07)	85.71(2.12)	86.56(1.98)	86.56(1.98)	87.35(1.34)	85.71(5.52)	96.20(1.72)
Wine	**96.94(4.08)**	95.00(1.53)	95.55(2.30)	95.85(2.25)	96.16(3.02)	96.66(3.36)	95.00(4.14)	100.00(0.21)
Haberman	**76.71(3.52)**	71.23(4.16)	72.86(3.65)	70.16(3.56)	72.26(4.17)	65.01(3.20)	75.00(3.40)	97.36(3.34)

In Table IV-1, we compare the recognition rates obtained by the proposed meta-learning framework against dynamic selection techniques explained in this paper: Overall Local Accuracy (OLA) [22], Local Classifier Accuracy (LCA) [22], Modified Local Accuracy (MLA) [29], KNORA-Eliminate [14], K-Nearest Output Profiles (KNOP) [16] and the Multiple Classifier Behavior (MCB) [21]. We compare each pair of results using the Kruskal-Wallis non-

[2] www.prtools.org

parametric statistical test with a 95% confidence interval. The results of the proposed framework over the Pima, Liver Disorders, Blood Transfusion, Vehicle, Sonar and Ionosphere datasets are statistically superior to the result of the best DES from the literature. For the other datasets, Breast, Banana and Lithuanian, the results are statistically equivalent.

4.2 Dissimilarity Analysis

In this section, we conduct a dissimilarity analysis between distinct DES techniques. The analysis is performed based on the difference between the level of competence $\delta_{i,j}$ estimated by each DES technique for a given base classifier c_i, for each query sample $\mathbf{x}_j$ (Section 2). The goal of the dissimilarity analysis is twofold: to understand the behavior of different DES techniques (i.e., whether or not the criterion used by DES techniques present a similar behavior), and in order to see which DES criterion is closer to the behavior of the criterion used by the ideal DES scheme (Oracle) for the estimation of the competence level of a base classifier.

Given 8 dynamic selection techniques, the first step of the dissimilarity analysis is to compute the dissimilarity matrix D. This matrix D is an 8×8 symmetrical matrix, where each element $d_{A,B}$ represents the dissimilarity between two different DES techniques, A and B. Given that $\delta^A_{i,j}$ and $\delta^B_{i,j}$ are the levels of competence of c_i in relation to $\mathbf{x}_j$ for the techniques A and B, respectively, the dissimilarity $d_{A,B}$ is calculated by the difference between $\delta^A_{i,j}$ and $\delta^B_{i,j}$ (Equation A IV-4).

$$d_{A,B} = \frac{1}{NM} \sum_{j=1}^{N} \sum_{i=1}^{M} \left(\delta^A_{i,j} - \delta^B_{i,j} \right)^2 \tag{A IV-4}$$

where N and M are the size of the validation dataset and the pool of classifiers, respectively.

For each dataset considered in this work, a dissimilarity matrix (e.g., D_{Pima}, D_{Liver}) is computed, with the mean dissimilarity values over 20 replications. Then, the average dissimilarity matrix $\bar{D}$ is obtained by computing the mean and standard deviation of the eleven dissimilarity matrices. Table IV-2 shows the average dissimilarity matrix $\bar{D}$. Both the average and the

standard deviation values are presented. Each line or column of the dissimilarity matrix can be seen as one axe in the $8th$ dimensional space. Each axe in this space represents the distance to a specific DES technique, for instance, the first axe represents the distance to the proposed meta-learning framework; the second represents the distance to the KNORA technique and so forth.

Table-A IV-2 The average dissimilarity matrix $\bar{D}$. The values are the mean and standard deviation computed over the eleven dissimilarity matrix

	Meta-Learning	**KNORA**	**MCB**	**LCA**	**OLA**	**MLA**	**KNOP**	**Oracle**
Meta-Learning	0	0.36(0.06)	0.46(0.15)	0.40(0.07)	0.36(0.06)	0.40(0.04)	0.53(0.08)	0.54(0.03)
KNORA	0.36(0.06)	0	0.89(0.06)	0.42(0.01)	0.44(0.01)	0.71(0.04)	0.74(0.11)	0.68(0.01)
MCB	0.46(0.15)	0.89(0.06)	0	0.58(0.01)	0.89(0.06)	1.06(0.07)	0.75(0.03)	0.72(0.08)
LCA	0.40(0.07)	0.42(0.01)	0.58(0.01)	0	0.42(0.01)	0.45(0.02)	0.31(0.04)	0.60(0.06)
OLA	0.36(0.06)	0.44(0.01)	0.89(0.06)	0.42(0.01)	0	0.71(0.04)	0.74(0.11)	0.68(0.11)
MLA	0.40(0.04)	0.71(0.04)	1.06(0.07)	0.45(0.02)	0.71(0.04)	0	0.54(0.01)	0.63(0.07)
KNOP	0.53(0.08)	0.74(0.11)	0.75(0.03)	0.31(0.04)	0.74(0.11)	0.54(0.01)	0	0.86(0.12)
Oracle	0.54(0.03)	0.68(0.01)	0.72(0.08)	0.60(0.06)	0.68(0.11)	0.63(0.07)	0.86(0.12)	0

4.2.1 Classifier Projection Space

The next step is to project the dissimilarity matrix $\bar{D}$ onto the Classifier Projection Space (CPS) for a better visualization of the relationship between all techniques. The CPS is an $\mathbb{R}^n$ space where each technique is represented as a point and the Euclidean distance between two techniques is equal to their dissimilarities [113]. Techniques that are similar to one another appear closer in the CPS while those with a higher dissimilarity are more distant. Thus, it is possible to obtain a spatial representation of the dissimilarity between all techniques. A two-dimensional CPS is used for better visualization. To obtain a two-dimensional CPS, a dimensionality reduction of the dissimilarity matrix $\bar{D}$ in the $\mathbb{R}^8$ to $\tilde{D}$ in the $\mathbb{R}^2$ is required. This reduction is performed using Sammon mapping [117]; that is, a non-linear Multidimensional Scaling (MDS) projection onto a lower dimensional space such that the distances are preserved [113; 117].

Given the dissimilarity matrix $\bar{D}$, a configuration X of m points in $\mathbb{R}^k, (k \leq m)$ is computed using a linear mapping, called classical scaling [117]. The process is performed through rota-

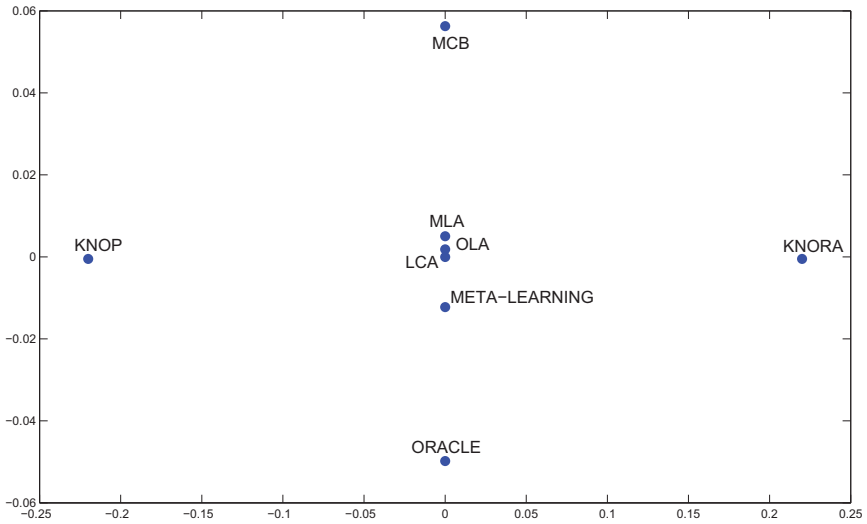
tion and translation, such that the distances after dimensionality reduction are preserved. The projection X is computed as follows: first, a matrix of the inner products is obtained by the square distances $B = -\frac{1}{2}JD^2J$, where $J = I - \frac{1}{m}UU^T$, and I and U are the identity matrix and unit matrix, respectively. J is used as a normalization matrix such that the mean of the data is zero. The eigendecomposition of B is then obtained as, $B = Q\Lambda Q^T$, where Λ is a diagonal matrix containing the eigenvalues (in decreasing order) and Q is the matrix of the corresponding eigenvectors. The configuration of points in the reduced space is determined by the k largest eigenvalues. Therefore, X is uncorrelated in the $\mathbb{R}^k$, $X = Q_k\sqrt{\Lambda_k}$ space. In our case, $k = 2$.

The CPS projection is obtained by applying Sammon mapping over the matrix X. The mapping is performed by defining a function, called stress function $\mathcal{S}$ (Equation A IV-5), which measures the difference between the original dissimilarity matrix $\bar{D}$ and the distance matrix of the projected configuration, $\tilde{D}$, where $\tilde{d}(i,j)$ is the distance between the classifiers i and j in the projection X.

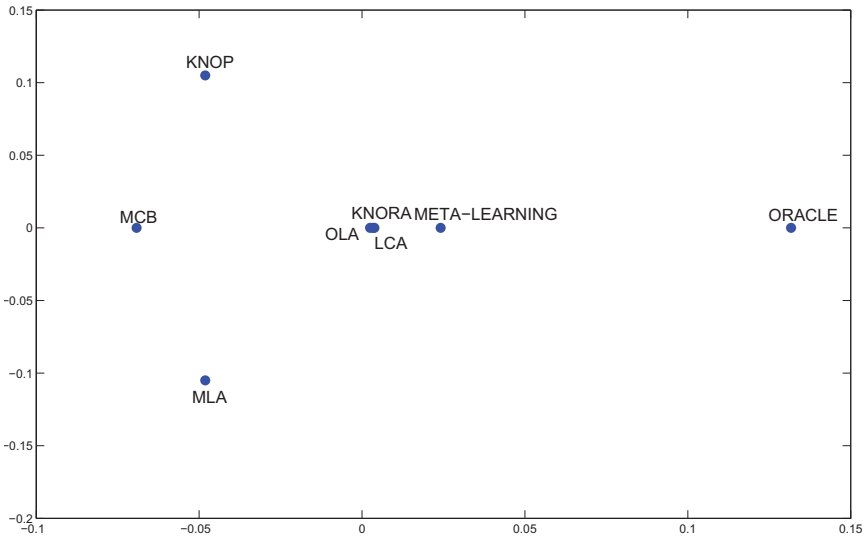
$$\mathcal{S} = \frac{1}{\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} d(i,j)^2} \sum_{i=1}^{m-1}\sum_{j=i+1}^{m} (d(i,j) - \tilde{d}(i,j)) \qquad \text{(A IV-5)}$$

The two-dimensional CPS plot is shown in Figure IV-2. Figure IV-2(a) shows the average CPS plot obtained considering the average dissimilarity matrix $\bar{D}$, while Figure IV-2(b) shows an example of the CPS plot obtained for the Liver Disorders dataset D_{Liver}.

An important observation that can be drawn from Figure IV-2(a) is that the LCA, OLA and MLA appear close together in the dissimilarity space. Which means, that the criteria used by these three techniques to estimate the level of competence of a base classifiers present similar behaviors when averaged over several classification problems. Thus, they are very likely to achieve the same results [11]. This can be explained by the fact that these three techniques are based on the same information (the classification accuracy over a defined local region in the feature space), with little difference regarding the use of a posteriori information by the LCA technique or weights for the MLA technique.



(a) CPS for the average dissimilarity matrix $\bar{D}$



(b) CPS for the dissimilarity matrix D_{Liver}, obtained for the Liver disorders dataset

Figure-A IV-2 Two-dimensional CPS plot for the average dissimilarity matrix $\bar{D}$ and for the dissimilarity matrix obtained for the Liver disorders dataset D_{Liver}. It is important to mention that the axes of the CPS plot cannot be interpreted alone. Only the Euclidean distances between the points count

The meta-learning framework appears closer to the Oracle in the two-dimensional CPS (Figures IV-2(a) and (b)). In addition, the meta-learning framework is also closer to the techniques

from the local accuracy paradigm (LCA, OLA and MLA) than to any other DES technique, which can be explained by the fact that three out of the five meta-features comes from estimations of the local regions (f_1, f_2 and f_3).

Table IV-3 presents the dissimilarity measure for each DES technique in relation to the Oracle. Results show that the proposed meta-learning framework is closer to the behavior of the Oracle as it presents the lowest dissimilarity value on average, 0.54. The LCA technique comes closer, with an average dissimilarity value of 0.60. Thus, we suggest that the use of multiple criteria to estimate the level of competence of a base classifier results in a DES technique that obtains a estimation of the level of competence of a base classifier closer to that provided by an ideal DES scheme (Oracle).

Table-A IV-3 Mean and standard deviation of the dissimilarity between each DES technique from the Oracle for each classification problem. The smallest dissimilarity values are highlighted

Database	Meta-Learning	KNORA-E	MCB	LCA	OLA	MLA	KNOP
Pima	**0.32(0.04)**	0.43(0.01)	0.47(0.08)	0.36(0.06)	0.43(0.01)	0.44(0.07)	0.41(0.02)
Liver Disorders	**0.50(0.04)**	0.61(0.01)	0.67(.008)	0.56(0.06)	0.61(0.01)	0.60(0.07)	0.51(0.02)
Breast Cancer	**0.59(0.35)**	1.22(0.10)	1.20(0.10)	0.69(0.01)	1.20(0.10)	0.77(0.03)	1.20(0.10)
Blood Transfusion	**0.33(0.03)**	0.40(0.01)	0.46(0.01)	0.36(.003)	0.40(0.01)	0.44(0.08)	0.4(0.01)
Banana	0.33(0.10)	0.29(0.01)	0.36(0.01)	**0.24(0.01)**	0.29(0.01)	0.36(0.01)	0.34(0.01)
Vehicle	**0.36(0.07)**	0.49(0.01)	0.48(0.02)	**0.36(0.04)**	0.49(0.01)	0.37(0.05)	0.47(0.02)
Lithuanian Classes	0.47(0.14)	0.49(0.02)	0.56(0.02)	**0.39(0.04)**	0.49(0.02)	0.54(0.01)	0.51(0.03)
Sonar	**0.58(0.10)**	0.91(0.04)	0.88(0.01)	0.70(0.01)	0.91(0.04)	0.85(0.02)	0.84(0.06)
Ionosphere	**0.62(0.22)**	0.89(0.05)	0.88(0.06)	0.70(0.07)	0.89(0.05)	0.68(0.02)	0.88(0.06)
Wine	1.03(0.20)	0.88(0.11)	0.98(0.11)	**0.73(0.02)**	0.88(0.11)	0.93(0.06)	0.82(0.14)
Haberman	**0.79(0.04)**	0.89(0.05)	1.01(0.05)	0.82(0.02)	0.89(0.05)	0.92(0.04)	0.86(0.06)
Mean	**0.54(0.05)**	0.68(0.01)	0.72(0.08)	0.60(0.06)	0.68(0.11)	0.63(0.07)	0.86(0.12)

5. Conclusion

In this paper, we conducted a study about the dissimilarity between different DES techniques. These dissimilarities are computed in order to generate a dissimilarity matrix. Through Sammon Mapping, the dissimilarity matrix is embedded in a two-dimensional space, called the

Classifier Projection Space (CPS), where the Euclidean distance between two feature representations reflects their dissimilarity.

Based on the visual representation provided by the CPS, we can draw two conclusions:

- The proposed technique is closer to the Oracle in the dissimilarity space, which indicates that the use of different types of information about the behavior of base classifiers is indeed necessary in order to achieve a DES technique that is closer to the Oracle.
- Techniques that use the same kind of information to compute the level of competence of the base classifiers, such as LCA, OLA and MLA, are more likely to present the same results when their performance is averaged over several problems.

Future works in this topic include: i) The design of new sets of meta-features; ii) Carrying out a comparison of different meta-features vectors in order to achieve a set of features that can better address the behavior of the Oracle; and, iii) Increasing the number of classification problems in the analysis.

BIBLIOGRAPHY

[1] A. S. Britto, R. Sabourin, L. E. S. de Oliveira, Dynamic selection of classifiers - A comprehensive review, Pattern Recognition 47 (11) (2014) 3665–3680.

[2] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, T. I. Ren, META-DES: A dynamic ensemble selection framework using meta-learning, Pattern Recognition 48 (5) (2015) 1925–1935.

[3] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140.

[4] T. K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 832–844.

[5] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Proceedings of the Second European Conference on Computational Learning Theory (1995) 23–37.

[6] L. I. Kuncheva, J. J. Rodríguez, Classifier ensembles with a random linear oracle, IEEE Trans. Knowl. Data Eng. 19 (4) (2007) 500–508.

[7] R. P. W. Duin, The combining classifier: to train or not to train?, Proceedings of the 16th International Conference on Pattern Recognition 2 (2002) 765–770.

[8] C. A. Shipp, L. I. Kuncheva, Relationships between combination methods and measures of diversity in combining classifiers, Information Fusion 3 (2002) 135–148.

[9] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.

[10] G. Giacinto, F. Roli, Design of effective neural network ensembles for image classification purposes, Image Vision Comput. 19 (9-10) (2001) 699–707.

[11] R. M. Cruz, G. D. Cavalcanti, I. R. Tsang, R. Sabourin, Feature representation selection based on classifier projection space and oracle analysis, Expert Systems with Applications 40 (9) (2013) 3813 – 3827.

[12] E. M. dos Santos, R. Sabourin, P. Maupin, Single and multi-objective genetic algorithms for the selection of ensemble of classifiers, Proceedings of the International Joint Conference on Neural Networks (2006) 3070–3077.

[13] I. Partalas, G. Tsoumakas, I. Vlahavas, Focused ensemble selection: A diversity-based method for greedy ensemble selection, Proceeding of the 18th European Conference on Artificial Intelligence (2008) 117–121.

[14] A. H. R. Ko, R. Sabourin, u. S. Britto, Jr., From dynamic classifier selection to dynamic ensemble selection, Pattern Recognition 41 (2008) 1735–1748.

[15] E. M. Dos Santos, R. Sabourin, P. Maupin, A dynamic overproduce-and-choose strategy for the selection of classifier ensembles, Pattern Recognition 41 (2008) 2993–3009.

[16] P. R. Cavalin, R. Sabourin, C. Y. Suen, Dynamic selection approaches for multiple classifier systems, Neural Computing and Applications 22 (3-4) (2013) 673–688.

[17] P. R. Cavalin, R. Sabourin, C. Y. Suen, Logid: An adaptive framework combining local and global incremental learning for dynamic selection of ensembles of HMMs, Pattern Recognition 45 (9) (2012) 3544–3556.

[18] T. Woloszynski, M. Kurzynski, A probabilistic model of classifier competence for dynamic ensemble selection, Pattern Recognition 44 (2011) 2656–2668.

[19] J. Xiao, L. Xie, C. He, X. Jiang, Dynamic classifier ensemble model for customer classification with imbalanced class distribution, Expert Systems with Applications 39 (2012) 3668–3675.

[20] R. M. O. Cruz, G. D. C. Cavalcanti, T. I. Ren, A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence, Proceedings of the International Joint Conference on Neural Networks (2011) 1126 – 1133.

[21] G. Giacinto, F. Roli, Dynamic classifier selection based on multiple classifier behaviour, Pattern Recognition 34 (2001) 1879–1881.

[22] K. Woods, W. P. Kegelmeyer, Jr., K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Transactions on Pattern Analysis Machine Intelligence 19 (1997) 405–410.

[23] L. Kuncheva, Switching between selection and fusion in combining classifiers: An experiment, IEEE Transactions on System, Man and Cybernetics 32 (2) (2002) 146–156.

[24] J. Kittler, M. Hatef, R. P. W. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 226–239.

[25] Y. Lu, Knowledge integration in a multiple classifier system., Applied Intelligence 6 (1996) 75–86.

[26] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts, Neural Computation 3 (1991) 79–87.

[27] X. Zhu, X. Wu, Y. Yang, Dynamic classifier selection for effective mining from noisy data streams, Proceedings of the 4th IEEE International Conference on Data Mining (2004) 305–312.

[28] S. Singh, M. Singh, A dynamic classifier selection and combination approach to image region labelling, Signal Processing:Image Communication 20 (3) (2005) 219–231.

[29] P. C. Smits, Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection, IEEE Transactions on Geoscience and Remote Sensing 40 (4) (2002) 801–813.

[30] L. I. Kuncheva, Clustering-and-selection model for classifier combination, Fourth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies (2000) 185–188.

[31] R. G. F. Soares, A. Santana, A. M. P. Canuto, M. C. P. de Souto, Using accuracy and diversity to select classifiers to build ensembles, Proceedings of the International Joint Conference on Neural Networks (2006) 1310–1316.

[32] L. I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2) (2002) 281–286.

[33] D. W. Corne, J. D. Knowles, No free lunch and free leftovers theorems for multiobjective optimisation problems, Evolutionary Multi-Criterion Optimization (EMO 2003) Second International Conference (2003) 327–341.

[34] T. K. Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 289–300.

[35] E. M. dos Santos, R. Sabourin, P. Maupin, A dynamic overproduce-and-choose strategy for the selection of classifier ensembles, Pattern Recognition 41 (10) (2008) 2993–3009.

[36] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, On meta-learning for dynamic ensemble selection, in: 22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014, 2014, pp. 1230–1235.

[37] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, A DEEP analysis of the META-DES framework for dynamic selection of ensemble of classifiers, CoRR abs/1509.00825.

[38] M. Sabourin, A. Mitiche, D. Thomas, G. Nagy, Classifier combination for handprinted digit recognition, Proceedings of the Second International Conference on Document Analysis and Recognition (1993) 163–166.

[39] K. M., W. T., R. Lysiak, On two measures of classifier competence for dynamic ensemble selection - experimental comparative analysis, in: International Symposium on Communications and Information Technologies, 2010, pp. 1108–1113.

[40] T. Woloszynski, M. Kurzynski, A measure of competence based on randomized reference classifier for dynamic ensemble selection, in: International Conference on Pattern Recognition (ICPR), 2010, pp. 4194–4197.

[41] B. Antosik, M. Kurzynski, New measures of classifier competence – heuristics and application to the design of multiple classifier systems., in: Computer recognition systems 4., 2011, pp. 197–206.

[42] R. G. F. Soares, A. Santana, A. M. P. Canuto, M. C. P. de Souto, Using accuracy and diversity to select classifiers to build ensembles, Proceedings of the International Joint Conference on Neural Networks (2006) 1310–1316.

[43] E. M. dos Santos, R. Sabourin, P. Maupin, Overfitting cautious selection of classifier ensembles with genetic algorithms, Information Fusion 10 (2) (2009) 150–162.

[44] L. Didaci, G. Giacinto, F. Roli, G. L. Marcialis, A study on the performances of dynamic classifier selection based on local accuracy estimation, Pattern Recognition 38 (11) (2005) 2188–2191.

[45] T. Woloszynski, M. Kurzynski, P. Podsiadlo, G. W. Stachowiak, A measure of competence based on random classification for dynamic ensemble selection, Information Fusion 13 (3) (2012) 207–213.

[46] T. Woloszynski, M. Kurzynski, On a new measure of classifier competence applied to the design of multiclassifier systems, in: International Conference on Image Analysis and Processing (ICIAP), 2009, pp. 995–1004.

[47] A. Wolczowski, M. Kurzynski, Human-machine interface in bioprosthesis control using EMG signal classification, Expert Systems 27 (1) (2010) 53–70.

[48] Y. S. Huang, C. Y. Suen, A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) 90–94.

[49] M. Aksela, Comparison of classifier selection methods for improving committee performance, in: Multiple Classifier Systems, 2003, pp. 84–93.

[50] G. Giacinto, F. Roli, Design of effective neural network ensembles for image classification purposes, Image and Vision Computing 19 (9-10) (2001) 699 – 707.

[51] M. C. P. de Souto, R. G. F. Soares, A. Santana, A. M. P. Canuto, Empirical comparison of dynamic classifier selection methods based on diversity and accuracy for building ensembles, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2008, pp. 1480–1487.

[52] R. Lysiak, M. Kurzynski, T. Woloszynski, Probabilistic approach to the dynamic ensemble selection using measures of competence and diversity of base classifiers, in: Hybrid Artificial Intelligent Systems - 6th International Conference, HAIS 2011, Wroclaw, Poland, May 23-25, 2011, Proceedings, Part II, 2011, pp. 229–236.

[53] J. Xiao, C. He, Dynamic classifier ensemble selection based on GMDH, in: Proceedings of the Second International Joint Conference on Computational Sciences and Optimization, CSO 2009, Sanya, Hainan, China, 24-26 April 2009, Volume 1, 2009, pp. 731–734.

[54] A. G. Ivakhnenko, Heuristic self-organization in problems of engineering cybernetics, Automatica 6 (2) (1970) 207–219.

[55] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, Analyzing dynamic ensemble selection techniques using dissimilarity analysis, in: Artificial Neural Networks in Pattern Recognition (ANNPR), 2014, pp. 59–70.

[56] J. H. Krijthe, T. K. Ho, M. Loog, Improving cross-validation based classifier selection using meta-learning, International Conference on Pattern Recognition (2012) 2873–2876.

[57] L. I. Kuncheva, J. C. Bezdek, R. P. W. Duin, Decision templates for multiple classifier fusion: an experimental comparison, Pattern Recognition 34 (2001) 299–314.

[58] M. Skurichina, R. P. W. Duin, Bagging for linear classifiers, Pattern Recognition 31 (1998) 909–930.

[59] K. Bache, M. Lichman, UCI machine learning repository (2013).
URL http://archive.ics.uci.edu/ml

[60] R. D. King, C. Feng, A. Sutherland, Statlog: Comparison of classification algorithms on large real-world problems (1995).

[61] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, Multiple-Valued Logic and Soft Computing 17 (2-3) (2011) 255–287.

[62] L. Kuncheva, Ludmila kuncheva collection LKC (2004).
URL http://pages.bangor.ac.uk/~mas00a/activities/real_data.htm

[63] R. P. W. Duin, P. Juszczak, D. de Ridder, P. Paclik, E. Pekalska, D. M. Tax, Prtools, a matlab toolbox for pattern recognition (2004).
URL http://www.prtools.org

[64] P. R. Cavalin, R. Sabourin, C. Y. Suen, Dynamic selection of ensembles of classifiers using contextual information, Multiple Classifier Systems (2010) 145–154.

[65] E. M. dos Santos, R. Sabourin, Classifier ensembles optimization guided by population oracle (2011) 693–698.

[66] D. Ruta, B. Gabrys, Classifier selection for majority voting, Information Fusion 6 (1) (2005) 63–81.

[67] G. Valentini, An experimental bias-variance analysis of svm ensembles based on resampling techniques, IEEE Transactions on Systems, Man, and Cybernetics, Part B 35 (2005) 1252–1271.

[68] P. Henniges, E. Granger, R. Sabourin, Factors of overtraining with fuzzy artmap neural networks, International Joint Conference on Neural Networks (2005) 1075–1080.

[69] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, META-DES.H: A dynamic ensemble selection technique using meta-learning and a dynamic weighting approach, in: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015, 2015, pp. 1–8.

[70] W. I. Ai, P. Langley, Induction of one-level decision trees, in: Proceedings of the Ninth International Conference on Machine Learning, Morgan Kaufmann, 1992, pp. 233–240.

[71] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, Journal of Machine Learning Research 15 (2014) 3133–3181.
URL http://jmlr.org/papers/v15/delgado14a.html

[72] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: The rprop algorithm, Proceedings of the IEEE International Conference on Neural Networks (1993) 586–591.

[73] R. N. Khushaba, A. Al-Ani, A. Al-Jumaily, Feature subset selection using differential evolution and a statistical repair mechanism, Expert Systems with Applications 38 (9) (2011) 11515–11526.

[74] R. L. Haupt, S. E. Haupt, Practical genetic algorithms, J. Wiley, Hoboken, N.J., 2004.

[75] P. V. W. Radtke, T. Wong, R. Sabourin, An evaluation of over-fit control strategies for multi-objective evolutionary optimization, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN, 2006, pp. 3327–3334.

[76] J. Kennedy, R. C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[77] S. Mirjalili, A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization, Swarm and Evolutionary Computation 9 (2013) 1–14.

[78] L.-Y. Chuang, S.-W. Tsai, C.-H. Yang, Improved binary particle swarm optimization using catfish effect for feature selection., Expert Systems with Applications 38 (10) (2011) 12699–12707.

[79] K. Price, R. M. Storn, J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[80] A. Al-Ani, A. Alsukker, R. N. Khushaba, Feature subset selection using differential evolution and a wheel based search strategy, Swarm and Evolutionary Computation 9 (2013) 15–26.

[81] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.

[82] M. Kudo, J. Sklansky, Comparison of algorithms that select features for pattern classifiers., Pattern Recognition 33 (1) (2000) 25–41.

[83] L.-Y. Chuang, H.-W. Chang, C.-J. Tu, C.-H. Yang, Improved binary pso for feature selection using gene expression data, Comput. Biol. Chem. 32 (1) (2008) 29–38.

[84] H. A. Firpi, E. D. Goodman, Swarmed feature selection, in: 33rd Applied Image Pattern Recognition Workshop (AIPR 2004), Emerging Technologies and Applications for Imagery Pattern Recognition, 13-15 October 2004, Washington, DC, USA, Proceedings, 2004, pp. 112–118.

[85] L. Wang, X. Wang, J. Fu, L. Zhen, A novel probability binary particle swarm optimization algorithm and its application, JSW 3 (9) (2008) 28–35.

[86] E. Alpaydin, M. I. Jordan, Local linear perceptrons for classification., IEEE Transactions on Neural Networks 7 (3) (1996) 788–794.

[87] E. M. dos Santos, L. S. Oliveira, R. Sabourin, P. Maupin, Overfitting in the selection of classifier ensembles: a comparative study between pso and ga., in: GECCO, ACM, 2008, pp. 1423–1424.

[88] L. Breiman, Prediction games and arcing algorithms, Neural Computation 11 (7) (1999) 1493–1517.

[89] S. Kullback, R. A. Leibler, On information and sufficiency, Annals of Mathematical Statistics 22 (1) (1951) 79–86.

[90] J. H. Friedman, L. C. Rafsky, The Annals of Statistics 7 (4) (1979) 697–717.

[91] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[92] G. Valentini, Ensemble methods based on bias-variance analysis, Ph.D. thesis, Dipartimento di Informatica e Scienze dell' Informazione (DISI) - Università di Genova (2003).

[93] D. L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man and Cybernetics 2 (3) (1972) 408–421.

[94] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435.

[95] M. R. Smith, T. R. Martinez, C. G. Giraud-Carrier, An instance level analysis of data complexity, Machine Learning 95 (2) (2014) 225–256.

[96] J. Wang, P. Neskovic, L. N. Cooper, Improving nearest neighbor rule with a simple adaptive distance measure, Pattern Recognition Letters 28 (2007) 207–213.

[97] J. Díez-Pastor, J. J. Rodríguez, C. García-Osorio, L. I. Kuncheva, Diversity techniques improve the performance of the best imbalance learning ensembles, Information Sciences 325 (2015) 98–117.

[98] L. Nanni, C. Fantozzi, N. Lazzarini, Coupling different methods for overcoming the class imbalance problem, Neurocomputing 158 (2015) 48 – 61.

[99] I. Triguero, J. Derrac, S. García, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, IEEE Trans. Systems, Man, and Cybernetics, Part C 42 (1) (2012) 86–100.

[100] J. Calvo-Zaragoza, J. Valero-Mas, J. Rico-Juan, Prototype generation on structural data using dissimilarity space representation, Neural Computing and Applications (2016) 1–10.

[101] D. Bertolini, L. S. Oliveira, E. J. R. Justino, R. Sabourin, Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers, Pattern Recognition 43 (1) (2010) 387–396.

[102] R. M. O. Cruz, G. D. C. Cavalcanti, T. I. Ren, An ensemble classifier for offline cursive character recognition using multiple feature extraction techniques, Proceedings of the International Joint Conference on Neural Networks (2010) 744–751.

[103] D. M. J. Tax, M. van Breukelen, R. P. W. Duin, J. Kittler, Combining multiple classifiers by averaging or by multiplying?, Pattern Recognition 33 (9) (2000) 1475–1485.

[104] R. M. O. Cruz, G. D. C. Cavalcanti, T. I. Ren, Handwritten digit recognition using multiple feature extraction techniques and classifier ensemble, 17th International Conference on Systems, Signals and Image Processing (2010) 215–218.

[105] N. Macià, E. Bernadó-Mansilla, A. Orriols-Puig, T. K. Ho, Learner excellence biased by data set selection: A case for data characterisation and artificial data sets, Pattern Recognition 46 (3) (2013) 1054–1066.

[106] M. H. Nguyen, H. A. Abbass, R. I. McKay, A novel mixture of experts model based on cooperative coevolution, Neurocomputing 70 (1-3) (2006) 155–163.

[107] M. Friedman, The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.

[108] O. D. Trier, A. K. Jain, T. Taxt, Feature extraction methods for character recognition: A survey, Pattern Recognition 29 (4) (1995) 641–662.

[109] L. S. Oliveira, R. Sabourin, F. Bortolozzi, C. Y. Suen, Automatic recognition of handwritten numerical strings: A recognition and verification strategy, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (11) (2002) 1438–1454.

[110] E. Kavallieratou, K. Sgarbas, N. Fakotakis, G. Kokkinakis, Handwritten word recognition based on structural characteristics and lexical support, Proceedings of the Seventh International Conference on Document Analysis and Recognition (2003) 562–567.

[111] Y. Chim, A. A. Kassim, Y. Ibrahim, Dual classifier system for handprinted alphanumeric character recognition, Pattern Analysis and Application 4 (1) (1998) 155–162.

[112] Z. Ping, C. Lihui, A novel feature extraction method and hybrid tree classification for handwritten numeral recognition, Pattern Recognition Letters 23 (1-3) (2002) 45–56. doi:http://dx.doi.org/10.1016/S0167-8655(01)00088-5.

[113] E. Pekalska, R. P. W. Duin, M. Skurichina, A discussion on the classifier projection space for classifier combining, Proceedings of the Third International Workshop on Multiple Classifier Systems (2002) 137–148.

[114] L. Schomaker, M. Bulacu, Automatic writer identification using connected-component contours and edge-based features of uppercase western script, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (6) (2004) 787–798. doi:http://dx.doi.org/10.1109/TPAMI.2004.18.

[115] S. N. Srihari, C. I. Tomai, B. Zhang, S. Lee, Individuality of numerals, Proceedings of the Seventh International Conference on Document Analysis and Recognition (2003) 1086–1091.

[116] L. I. Kuncheva, C. J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2) (2003) 181–207.

[117] T. F. Cox, M. A. A. Cox, Multidimensional Scaling, 2nd Edition, Chapman and Hall, 2000.

[118] E. Pekalska, R. P. W. Duin, Spatial representation of dissimilarity data via lower-complexity linear and nonlinear mappings, in: Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Springer-Verlag, 2002, pp. 488–497.

[119] L. I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Transactions on Pattern Analysis Machine Intelligence 24 (2002) 281–286.

[120] P. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, London, 1982.

[121] P. Zhang, T. D. Bui, C. Y. Suen, A novel cascade ensemble classifier system with a high recognition performance on handwritten digits, Pattern Recognition 40 (12) (2007) 3415–3429.

[122] F. Camastra, A SVM-based cursive character recognizer, Pattern Recognition 40 (12) (2007) 3721–3727. doi:http://dx.doi.org/10.1016/j.patcog.2007.03.014.

[123] R. C. Gonzalez, R. E. Woods, Digital Image Processing, 3rd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[124] S. Impedovo, M. G. Lucchese, G. Pirlo, Optimal zoning design by genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics, Part A 36 (5) (2006) 833–846.

[125] T. Y. Zhang, C. Y. Suen, A fast parallel algorithm for thinning digital patterns, Communications of the ACM 27 (3) (1984) 236–239.

[126] F. Camastra, M. Spinetti, A. Vinciarelli, Offline cursive character challenge: a new benchmark for machine learning and pattern recognition algorithms., Proceedings of the 18th International Conference on Pattern Recognition (2006) 913–916.

[127] J. Thornton, M. Blumenstein, V. Nguyen, T. Hine, Offline cursive character recognition: A state-of-the-art comparison, 14th Conference of the International Graphonomics Society (2009) 148–152.

[128] M. D. Richard, R. P. Lippmann, Neural network classifiers estimate bayesian a posteriori probabilities, Neural Computation 3 (4) (1991) 461–483.
URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1991.3.4.461

[129] L. I. Kuncheva, Switching between selection and fusion in combining classifiers: an experiment, IEEE Transactions on Systems, Man, and Cybernetics, Part B 32 (2) (2002) 146–156.

[130] Y. Xu, X. Cao, H. Qiao, An efficient tree classifier ensemble-based approach for pedestrian detection, IEEE Transactions on Systems, Man, and Cybernetics, Part B 41 (1) (2011) 107–117.

[131] B. Scholkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[132] J. Thornton, J. Faichney, M. Blumenstein, T. Hine, Character recognition using hierarchical vector quantization and temporal pooling, Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence (2008) 562–572.

[133] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[134] F. Lauer, C. Y. Suen, G. Bloch, A trainable feature extractor for handwritten digit recognition, Pattern Recognition 40 (6) (2007) 1816–1824. doi:http://dx.doi.org/10.1016/j.patcog.2006.10.011.

[135] P. Y. Simard, D. Steinkraus, J. C. Platt, Best practices for convolutional neural networks applied to visual document analysis, International Conference on Document Analysis and Recognition 2 (2003) 958–963.

[136] M. Ranzato, Y.-L. Boureau, Y. LeCun, Sparse feature learning for deep belief networks, Advances in Neural Information Processing Systems (2008) 1185–1192.

[137] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition? (2009) 2146–2153.

[138] D. C. Ciresan, U. Meier, L. M. Gambardella, J. Schmidhuber, Deep big simple neural nets excel on handwritten digit recognition, CoRR abs/1003.0358.

[139] E. Kussul, T. Baidyk, D. Wunsch, O. Makeyev, A. Martin, Permutation coding technique for image recognition systems, IEEE Transactions on Neural Networks 17 (6) (2006) 1566–1579.

[140] B. E. Labusch, K., T. Martinetz, Simple method for high-performance digit recognition based on sparse coding, IEEE Transactions on Neural Networks 19 (11) (2008) 1985–1989.

[141] P. Zhang, Reliable recognition of handwritten digits using a cascade ensemble classifier system and hybrid features, Ph.D. thesis, Concordia University, Montreal, P.Q., Canada (2006).